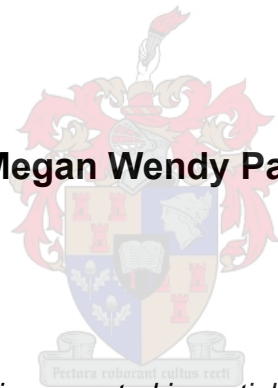


# **Bayesian Machine Learning: Theory and Applications**

**Megan Wendy Payne**



*Thesis presented in partial fulfilment  
of the requirements for the degree of  
Master of Commerce (Mathematical Statistics)  
in the Faculty of Economic and Management Sciences at Stellenbosch University*

**Supervisor: Dr Justin Harvey**

**December 2020**

## Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Initials and surname	Date
M W Payne	December 2020

## **Acknowledgements**

I hereby would like to acknowledge my supervisor, Dr Justin Harvey, as well as the entire department of Statistics and Actuarial Science at Stellenbosch University for making this thesis possible. Secondly, I would like to acknowledge and thank my family for their endless support. Lastly, I would like to thank my fiancé – my number one fan.

## Abstract

Machine learning problems in general are concerned with the ability of different methods and algorithms to extract useful and interpretable information from large datasets, possibly ones which are corrupt due to noisy measurements or errors in data capturing. As the size and complexity of data increases, the demand for efficient and robust machine learning techniques is greater than ever. All statistical techniques can be divided into either a frequentist approach or a Bayesian approach depending on how probability is interpreted and how the unknown parameter set is treated.

Bayesian methods have been present for several centuries; however, it was the advent of improved computational power and memory storage that catalysed the use of Bayesian modelling approaches in a wider range of scientific fields. This is largely due to many Bayesian methods requiring the computation of complex integrals, sometimes ones that are analytically intractable to compute in closed form, now being more accessible for use since approximation methods are less time-consuming to execute.

This thesis will consider a Bayesian approach to statistical modelling and takes the form of a postgraduate course in Bayesian machine learning. A comprehensive overview of several machine learning topics are covered from a Bayesian perspective and, in many cases, compared with their frequentist counterparts as a means of illustrating some of the benefits that arise when making use of Bayesian modelling. The topics covered are focused on the more popular methods in the machine learning literature.

Firstly, Bayesian approaches to classification techniques as well as a fully Bayesian approach to linear regression are discussed. Further, no discussion on machine learning methods would be complete without consideration of variable selection techniques, thus, a range of Bayesian variable selection and sparse Bayesian learning methods are considered and compared. Finally, probabilistic graphical models are presented since these methods form an integral part of Bayesian artificial intelligence.

Included with the discussion of each technique is a practical implementation. These examples are all easily reproducible and demonstrate the performance of each method. Where applicable, a comparison of the Bayesian and frequentist methods are provided. The topics covered are by no means exhaustive of the Bayesian machine learning literature but rather provide a comprehensive overview of the most commonly encountered methods.

**Key words:** Bayesian, classification, machine learning, probabilistic graphical models, regression, variable selection

## Opsomming

Masjienleer probleme het oor die algemeen te make met die vermoë van verskillende metodes en algoritmes om nuttige en interpreteerbare inligting uit groot en moontlik onbruikbare datastelle te haal. Soos die grootte en kompleksiteit van data toeneem, is die aanvraag vir doeltreffende en robuuste masjienleertegnieke groter as ooit tevore. Alle statistiese tegnieke kan in 'n frekwentistiese of 'n Bayes-benadering verdeel word, afhangende van hoe die waarskynlikheid geïnterpreteer word en hoe die onbekende parameterstel hanteer word.

Bayes metodes bestaan al 'n hele paar dekades lank. Dit was egter die koms van verbeterde rekenaarkrag en geheue-berging wat die gebruik van Bayes-modelleringsbenaderings in 'n wyer verskeidenheid wetenskaplike velde gekategoriseer het. Dit is grotendeels te danke aan baie Bayes-metodes wat die berekening van komplekse integrale vereis, wat soms analities onuitvoerbaar is om in geslote vorm te bereken, wat nou meer toeganklik is vir gebruik, aangesien benaderingsmetodes minder tydrowend is om uit te voer.

In hierdie proefskrif word die Bayes-benadering tot statistiese modellering bespreek en is in die vorm van 'n nagraadse kursus in Bayes-masjienleer. 'n Omvattende oorsig van verskeie masjienleeronderwerpe word vanuit 'n Bayes-perspektief behandel. In baie gevalle word dit vergelyk met hul frekwentistiese-eweknieë om die voordele van Bayes-modellering gebruik word, te illustreer. Die onderwerpe wat behandel word, fokus op die meer gewilde metodes in die masjienleerliteratuur.

Eerstens word Bayes-benaderings tot klassifikasietegnieke sowel as 'n volledige Bayes-benadering tot lineêre regressie bespreek. Verder sou geen bespreking oor masjienleermetodes volledig wees sonder inagneming van tegnieke vir veranderlike seleksie nie. 'n Reeks Bayes veranderlike seleksie en sommige Bayes-leermetodes word dus oorweeg en vergelyk. Laastens word grafiese waarskynlikheidsmodelle bespreek, aangesien hierdie metodes 'n belangrike rol in Bayes kunsmatige intelligensie speel.

'n Praktiese voorbeeld is by die bespreking van elke tegniek ingesluit. Hierdie voorbeelde is maklik om te hergebruik en wys die voordele van elke metode. Waar moontlik, word ook 'n vergelyking van die Bayes en frekwentistiese-metodes gegee. Die onderwerpe wat aangebied word, sluit geensins die volledig Bayes-masjienleerliteratuur in nie, maar bied 'n omvattende oorsig van die metodes wat die meeste voorkom en gebruik word.

**Sleutelwoorde:** Bayes, grafiese waarskynlikheidsmodelle, klassifikasietegnieke, masjienleerregressie, regressie, veranderlike seleksie,

## Table of contents

<b>Acknowledgements</b>	<b>iii</b>
<b>Abstract</b>	<b>iv</b>
<b>Opsomming</b>	<b>v</b>
<b>List of figures</b>	<b>x</b>
<b>List of tables</b>	<b>xii</b>
<b>List of appendices</b>	<b>xiii</b>
<b>List of abbreviations and/or acronyms</b>	<b>xiv</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 INTRODUCTION	1
1.2 MACHINE LEARNING	1
1.2.1 The Bias-Variance Trade-Off	2
1.3 BAYESIAN MODELLING	3
1.4 APPLICATIONS	5
1.5 CHAPTER OUTLINES	6
<b>CHAPTER 2 PROBABILITY THEORY AND MONTE CARLO METHODS</b>	<b>8</b>
2.1 PROBABILITY THEORY	8
2.1.1 Bayes' Theorem	9
2.1.2 The Prior Distribution	10
2.1.3 The Multinomial Distribution	10
2.1.4 The Dirichlet Distribution	13
2.2 MONTE CARLO METHODS	13
2.2.1 Markov Chains	14
2.2.2 Markov Chain Monte Carlo	15
2.2.2.1 <i>The Metropolis-Hastings Algorithm</i>	16
2.2.2.2 <i>Gibbs Sampling</i>	19
2.2.2.3 <i>Reversible Jump MCMC</i>	20
2.3 APPROXIMATE BAYESIAN COMPUTATION	22

2.4	SUMMARY	25
<b>CHAPTER 3 BAYESIAN APPROACHES TO CLASSIFICATION</b>		<b>26</b>
3.1	DISCRIMINATE VERSUS GENERATIVE MODELS	26
3.2	LOGISTIC REGRESSION	27
3.2.1	Bayesian Logistic Regression	30
3.2.2	Laplace Approximation	30
3.2.3	Applications	36
3.2.3.1	<i>Simulated Data</i>	36
3.2.3.2	<i>South African Heart Disease Data</i>	38
3.3	NAÏVE BAYES	41
3.3.1	Maximum Likelihood Estimation	42
3.3.2	Naïve Bayes Variants	45
3.3.3	Application: Sentiment Analysis	47
3.3.4	Bayesian Naïve Bayes	48
3.5	SUMMARY	54
<b>CHAPTER 4 A BAYESIAN APPROACH TO LINEAR REGRESSION</b>		<b>56</b>
4.1	CLASSICAL MULTIPLE LINEAR REGRESSION	56
4.2	BAYESIAN LINEAR REGRESSION	56
4.2.1	The Prior vs The Likelihood	60
4.2.2	The MAP Estimator	65
4.2.3	The Predictive Distribution	66
4.3	THE FULLY BAYESIAN APPROACH	70
4.3.1	Bayesian Model Selection	71
4.3.2	Type-II Maximum Likelihood (ML-II)	76
4.3.2.1	<i>The EM Algorithm</i>	77
4.4	SUMMARY	82
<b>CHAPTER 5 BAYESIAN VARIABLE SELECTION</b>		<b>83</b>
5.1	THE VARIABLE SELECTION PROBLEM	83

5.2	ZELLNER'S G-PRIORS	84
5.3	THE SPIKE-AND-SLAB PRIOR	94
5.4	THE BAYESIAN LASSO & RIDGE REGRESSION	97
5.4.1	A Bayesian Perspective of Ridge Regression	97
5.4.2	The Bayesian LASSO	99
5.5	APPLICATIONS	103
5.6	THE RELEVANCE VECTOR MACHINE	110
5.6.1	Support Vector Machines	111
5.6.2	Relevance Vector Regression	114
5.6.3	Relevance Vector Classification	122
5.7	SUMMARY	125
	<b>CHAPTER 6 PROBABILISTIC GRAPHICAL MODELS</b>	<b>126</b>
6.1	GRAPH THEORY	126
6.2	DIRECTED GRAPHICAL MODELS	126
6.2.1	Bayesian Belief Networks	126
6.2.2	Markov Chains	133
6.2.3	Dynamic Bayesian Networks	134
6.2.3.1	<i>Hidden Markov Models</i>	135
6.2.3.2	<i>Linear Dynamical Systems</i>	138
6.2.3.3	<i>Particle Filters</i>	142
6.3	UNDIRECTED GRAPHICAL MODELS	145
6.3.1	Markov Networks	145
6.3.2	Conditional Random Fields	150
6.4	SUMMARY	152
	<b>CHAPTER 7 CONCLUSION</b>	<b>153</b>
	<b>REFERENCES</b>	<b>155</b>
	<b>APPENDIX A CHAPTER 2 CODE</b>	<b>161</b>
	<b>APPENDIX B CHAPTER 3 CODE</b>	<b>167</b>



<b>APPENDIX C CHAPTER 4 CODE</b>	<b>181</b>
<b>APPENDIX D CHAPTER 5 CODE</b>	<b>189</b>
<b>APPENDIX E CHAPTER 6 CODE</b>	<b>207</b>

## List of figures

Figure 1.1	Visual representation of the bias-variance trade-off
Figure 1.2	Schematic overview of the various topics covered
Figure 2.1	Metropolis-Hastings output for the Gaussian simulation example
Figure 2.2	Trace plots for each variable in the bivariate normal model
Figure 2.3	Simulated data using the estimates obtained from the Gibbs sampling algorithm (left) and the true bivariate density function (right)
Figure 2.4	Trace plots obtained from reversible jump MCMC
Figure 2.5	Results of the ABC rejection sampling algorithm
Figure 3.1	Logistic Sigmoid function
Figure 3.2	Laplace approximation of a beta density function (left) and a Gaussian mixture model (right)
Figure 3.3	Simulated dataset
Figure 3.4	The decision boundary using a point estimate (left) and the decision boundary using MC averaging (right)
Figure 3.5	Posterior distributions of the parameters using MCMC
Figure 3.6	Histograms of the MCMC samples for each parameter overlaid with the marginal density function for each parameter obtained using Laplace approximation
Figure 3.7	Word clouds for positive sentiments (left) and the negative sentiments (right)
Figure 4.1	Simulated data from the linear model
Figure 4.2	Bayesian regression using one data observation
Figure 4.3	Bayesian regression with sequential inclusion of observed data points
Figure 4.4	Comparison of informative and non-informative priors
Figure 4.5	Posterior and predictive distributions for the simulated data
Figure 4.6	Model evidence for different degree polynomials
Figure 4.7	Results of the EM algorithm
Figure 4.8	Fully Bayesian approach to the linear regression problem
Figure 5.1	Simulated example of Lindley's paradox and the information paradox

Figure 5.2	Dirac and absolutely continuous spikes for a spike-and-slab prior
Figure 5.3	Gaussian and Laplace prior distributions in two dimensions
Figure 5.4	Approximate posterior distributions obtained from MCMC
Figure 5.5	Inclusion probabilities (left) and trace plots (right) for each parameter using a spike-and-slab prior
Figure 5.6	Box plots of the posterior distributions for the Bayesian LASSO and Bayesian ridge regression using reversible jump MCMC (top) and without reversible jump MCMC (bottom)
Figure 5.7	Optimal separating hyperplane (a), support vector classifier (b) and support vector classifier (c) for the simulated linearly separable data
Figure 5.8	Support vector regression (top) and relevance vector regression (bottom) applied to the simulated data
Figure 5.9	Support vector machine (left) and relevance vector classifier (right)
Figure 6.1	Fully connected, directed, acyclic graph (left) and the same graph but containing conditional independencies (right)
Figure 6.2	Bayesian network representation of the Naïve bayes model from Section 3.3
Figure 6.3	Bayesian network representation of the Asia dataset
Figure 6.4	Joint distribution of two variables represented as a Bayesian network
Figure 6.5	Hidden Markov model fit to the dice rolling problem
Figure 6.6	24 random digits of the Semeion Handwritten Digit dataset
Figure 6.7	Output of the Kalman filtering algorithm applied to a simple random walk model
Figure 6.8	Visual representation of a general particular filter
Figure 6.9	Robot localization using particle filters
Figure 6.10	Markov random field applied to the image denoising problem
Figure 6.11	Visual representation of the undirected graphical model for the image denoising problem

## List of tables

Table 1.1	Overview of R packages used
Table 2.1	Summary of notation
Table 3.1	Parameter coefficient estimates
Table 3.2	Test errors obtained using the three approaches to logistic regression
Table 3.3	Comparison of raw test review and cleaned text review
Table 3.4	Confusion matrix for the Bernoulli Naïve Bayes classifier
Table 3.5	Class conditional probabilities
Table 3.6	Confusion matrix for the Bayesian naïve Bayes model
Table 3.7	Class conditional probabilities for Bayesian naïve Bayes
Table 4.1	Changes in posterior predictive variance for increased sample size
Table 5.1	Inclusion probabilities for the two variants of Zellner's $g$ -prior
Table 5.2	Posterior medians and 95% credible intervals for each parameter using the two variants of Zellner's $g$ -prior
Table 5.3	Posterior median and 95% credible interval for each parameter using a spike-and-slab prior
Table 5.4	Posterior medians and 95% credible intervals for Bayesian ridge regression with and without reversible jump MCMC
Table 5.5	Posterior medians and 95% credible intervals for the Bayesian LASSO with and without reversible jump MCMC
Table 5.6	Comparison of test error, standard error and number of variables in the final model for each of the variable selection techniques
Table 5.7	Popular kernel functions for support vector machines
Table 6.1	Conditional probability table for the TB node
Table 6.2	Example of the data matrix for sentence number 28 in the NER dataset

## **List of appendices**

APPENDIX A	Chapter 2 code
APPENDIX B	Chapter 3 code
APPENDIX C	Chapter 4 code
APPENDIX D	Chapter 5 code
APPENDIX E	Chapter 6 code

## List of abbreviations and/or acronyms

ABC	Approximate Bayesian computation
ARD	Automatic relevance determination
BN	Bayesian belief network/Bayes Net
CPT	Conditional probability table
CRF	Conditional random field
EB	Empirical Bayes
EM	Expectation-maximization
GM	Graphical model
HMM	Hidden Markov model
HPM	Highest posterior probability model
i.i.d	Independently and identically distributed
IRLS	Iteratively reweighted least squares
LARS	Least Angle Regression
LASSO	Least Absolute Shrinkage and Selection Operator
LDS	Linear dynamical system
MAP	Maximum <i>a posteriori</i>
MCMC	Markov chain Monte Carlo
ML	Maximum likelihood
MLE	Maximum likelihood estimation/estimator
MPM	Median probability model
MRF	Markov random field
NER	Named entity recognition
OSH	Optimal separating hyperplane
PGM	Probabilistic graphical model
RJ	Reversible jump
RVM	Relevance vector machine
SLAM	Simultaneous localization and mapping

SVC	Support vector classifier
SVM	Support vector machine
TB	Tuberculosis

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION

The age of improved computing power and increased memory capacity of machines proved to be a catalyst for machine learning and big data analysis; this is even truer in the Bayesian machine learning paradigm since it allowed methods requiring computationally intensive estimations to be more realistic (and timeous) to execute.

This thesis focuses on Bayesian machine learning techniques and their applications in real world problems. The layout takes the form of a proposed postgraduate course in Bayesian machine learning. For this reason, a range of popular machine learning approaches from a Bayesian perspective are discussed, the necessary results derived in detail and the theory implemented using easily reproducible practical examples. The topics covered are by no means exhaustive, but rather illustrative of some of the more commonly encountered Bayesian machine learning techniques.

### 1.2 MACHINE LEARNING

The focus of any machine learning task is to learn from the data – the desired form of learning is what leads to different fields of study and applications. Machine learning tasks fall into two main categories: supervised and unsupervised learning problems. The most basic differentiation between supervised and unsupervised learning is that in supervised learning we have a response variable related in some way to the input, or predictor, variables, whereas in unsupervised learning, we only have input variables and no response. In supervised learning problems, the aim is typically either to predict the values of the response variable when a new set of data arrives, or to model the change in the response due to changes in the predictors, i.e. prediction or inference. In unsupervised learning, since there is no response variable available, the aim shifts to inferring properties of the underlying distribution of the predictors (based on the observed training sample) to identify any patterns or associations between them. A slightly less common category of machine learning is known as reinforcement learning (Murphy, 2012:2). This type of learning evolves through the use of reward and punishment signals and the process attempts to achieve some goal based on these signals.

Besides simply modelling these patterns and associations, a major field of research in machine learning is to develop algorithms that can be used efficiently and without too much dependence on computing capability other than that which a standard researcher would have at his or her disposal. Further, many datasets of interest for analysis tend to be highly corrupted with noise



and often contain missing values. Hence, algorithms need to be more than just efficient – they need to be robust and flexible enough to handle these issues without hampering the quality of their performance.

Machine learning techniques are found ubiquitously in the scientific literature, especially since the explosion of data availability and data accessibility. The methods and algorithms discussed in this thesis have foundations in statistical learning, medical diagnostics, robotics, and image processing, to name but a few. Our dependence on these models in everyday life prompts the need for continued research and development in machine learning to create improved, more efficient, and more accurate outcomes.

### 1.2.1 The Bias-Variance Trade-Off

One of the most common issues that arise in any machine learning problem is the trade-off between obtaining a model that is complex enough to capture all the necessary properties in the data, but also not so complex that training becomes slow and interpretability suffers. This issue is referred to as the bias-variance trade-off.

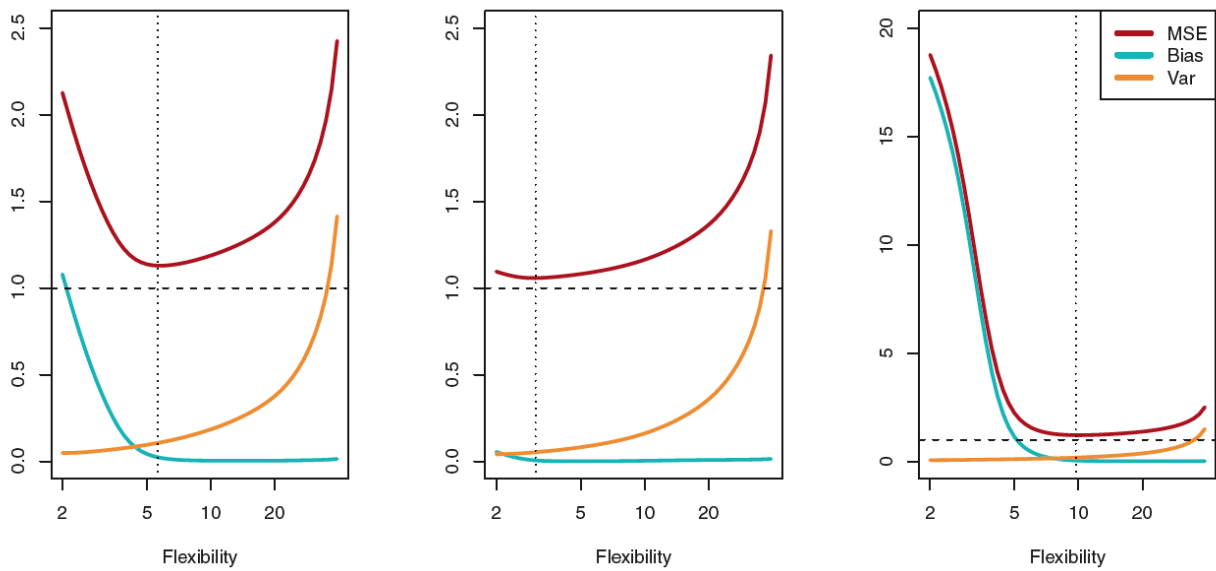
Given a training dataset, it will usually be possible to construct a model that fits the data almost perfectly, yielding a training error of zero. Naturally, this will be a very complex model with a large number of parameters. Due to this, the model will yield a very small bias from its close fit to the data, but a high variance due to overfitting and this model will almost surely perform worse on a new dataset. On the other hand, we can obtain a very low variance if we fit a simple model, such as a straight line, but this will result in a high bias.

This trade-off can conveniently be expressed mathematically if we make use of the squared error loss function (Theodoridis, 2015: 77). We consider a dataset where the response variable  $y$  can be modelled as  $y = f + \varepsilon$  where  $\varepsilon$  denotes the irreducible error with  $E[\varepsilon] = 0$  and  $Var[\varepsilon] = \sigma_f^2$ . Let the fitted model be denoted by  $\hat{f}$  where  $E[\hat{f}] = f$ . Using squared error loss, the associated test error is given by the expected mean squared error which can be decomposed as follows:

$$\begin{aligned}
 MSE &= E[(y - \hat{f})^2] \\
 &= E[(y - f + f - \hat{f})^2] \\
 &= E[(y - f)^2 + (f - \hat{f})^2 + 2(y - f)(f - \hat{f})] \\
 &= E[(f + \varepsilon - f)^2] + E[(f - \hat{f})^2] + 2E[(y - f)(f - \hat{f})] \\
 &= E[\varepsilon^2] + E(f - \hat{f})^2 + 0 \\
 &= E[\varepsilon^2] + E[(f - E[\hat{f}] - \hat{f} + E[\hat{f}])^2]
 \end{aligned}$$

$$\begin{aligned}
&= E[\varepsilon^2] + E \left[ (f - E[\hat{f}])^2 + (\hat{f} - E[\hat{f}])^2 + 2(f - E[\hat{f}])(\hat{f} - E[\hat{f}]) \right] \\
&= E[\varepsilon^2] + E \left[ (f - E[\hat{f}])^2 \right] + E \left[ (\hat{f} - E[\hat{f}])^2 \right] + 2E[(f - E[\hat{f}])(\hat{f} - E[\hat{f}])] \\
&= E[\varepsilon^2] + E \left[ (f - E[\hat{f}])^2 \right] + E \left[ (\hat{f} - E[\hat{f}])^2 \right] + 0 \\
&= \sigma_f^2 + \text{Bias}(\hat{f})^2 + \text{Var}(\hat{f}).
\end{aligned} \tag{1.1}$$

Hence, we see that the expected test error is the sum of the squared bias and the variance of the fitted model, plus the irreducible error (noise). Thus, for fixed values of the expected mean squared error, there is an inverse relationship that exists between the bias and the variance. This trade-off is displayed visually in Figure 1.1.



**Figure 1.1: Visual representation of the bias-variance trade-off.**

Source: James *et al.*, 2013:36

The vertical dotted gray line represents the optimal model complexity that will minimize the overall error in the model and the horizontal dashed line represents the irreducible error which is the smallest possible test error that can be obtained. As a result of this bias-variance trade-off, a large part of fitting machine learning models is determining what model, and what complexity of model, will result in the lowest overall test error rate. Bayesian model comparison and selection will be looked at in more detail in Section 4.3.1.

### 1.3 BAYESIAN MODELLING

All statistical analyses can be approached in one of two ways; either using a frequentist approach or a Bayesian approach. The differences in the approaches arise from the way in which each interprets probability. A frequentist interpretation of probability is based on the long run frequency

of events. That is, the probability of an event  $E$  occurring will be defined from a frequentist perspective as the limit

$$P(E) = \lim_{N \rightarrow \infty} \frac{N_E}{N} \quad (1.2)$$

where  $N_E$  denotes the number of times that event  $E$  occurs when a total number of  $N$  trials are performed. (Theodoridis, 2015:11). For example, when flipping an unbiased coin, the probability of obtaining heads on each flip is 0.5 since we will obtain heads approximately 50% of the time if we flip a fair coin a large number of times and record each outcome.

On the other hand, a Bayesian interpretation of probability is much more general. From a Bayesian perspective, probabilities are viewed as quantifications of uncertainty about an event. Hence, we begin with some initial beliefs about the state of a system or event. After obtaining new information about the event (i.e. by collecting a sample), we update our beliefs by incorporating this new information into our model. As a result, we have now gained more certainty about the event which we wish to model. This sequential update of beliefs takes place each time new data arrives and the error in estimation will be reduced with each update.

Both approaches rely on the likelihood function of the data to obtain parameter estimates. However, the fundamental difference in these approaches is the manner in which the parameter set itself is treated. In a frequentist approach, the parameters are defined as fixed, unknown values which are quantified via an estimator using methods such as maximum likelihood estimation (MLE). Hence, a point estimate is obtained for each of the parameters in the model. This point estimate does not inherently have any form of uncertainty associated with it. The uncertainty would need to be estimated by considering the distribution of different sample datasets that could be obtained. One of the most popular methods of obtaining estimates of standard errors for point estimates is using the bootstrap (Efron, 1979). On the other hand, in Bayesian inference, parameters of interest are treated as random variables which are assumed to follow some underlying distribution. The benefit of this approach is that the uncertainty in the parameter estimates can immediately be quantified through inspection of this distribution and interpretation during inference is more intuitive.

There are advantages and disadvantages that arise using either approach. For example, the frequentist approach of obtaining only point estimates means that the uncertainty in the estimate is not immediately included in the analysis. Further, point estimates can sometimes be very unreliable, especially when the data is multimodal or contains large spikes in areas far from the majority of the data points. Also, sole reliance on the likelihood function can lead to possibly inaccurate and misleading conclusions (Bishop, 2006:23). For example, we consider again the scenario of tossing a fair coin and the aim of modelling the probability of the coin showing up heads. If we base the analysis on, say, five coin tosses, and happen to obtain five consecutive

heads, then maximum likelihood estimation for this event would result in the estimate of the probability of heads occurring being equal to 1. On the other hand, if we had included prior knowledge reflecting (even vaguely) the belief of fairness of the coin, we would not obtain such an extreme conclusion.

One of the more common criticisms of a Bayesian approach is due to the fact that the choice of prior distribution is typically subjective and is sometimes chosen simply on the basis of computational ease, particularly to obtain closed form solutions for the posterior distributions and eliminate the need for sampling methods. Even if the prior distributions are reduced to being less informative, this can sometimes still lead to poor conclusions which highlights a major issue with Bayesian inference: poor conclusions can be easily be drawn with fairly high confidence as a result of ill-suited prior distributions.

Bayesian approaches started to gain significantly more traction in the machine learning literature after the introduction of sampling and approximation algorithms – such as Markov Chain Monte Carlo (Chapter 2) – which themselves became more feasible to implement as computers began to make substantial improvements. These improvements relate particularly to speed and memory capabilities. This was a catalyst in the Bayesian literature and has subsequently prompted a surge in Bayesian machine learning.

## **1.4 APPLICATIONS**

To aid explanations as well as demonstrate the uses of Bayesian machine learning techniques in practice, each of the sections discussed will have an associated practical implementation. All examples and applications done throughout this thesis have either been performed using simulated data or using freely available datasets. The freely available datasets were either obtained from the respective R packages provided in Table 1.1, or were obtained online from the UCI Machine Learning Repository (Dua & Graff, 2019) or from Kaggle<sup>1</sup>.

The coding used for all analyses can be found in each of the respective appendices. Therefore, all results and figures provided can easily be reproduced. The coding was performed in R and several different packages were used and these packages and where they were applied are summarized in Table 1.1. Only those packages that are specific to Bayesian modelling, or were the source of a dataset, have been included in this table.

---

<sup>1</sup> <http://www.kaggle.com>

**Table 1.1: Overview of R packages used.**

R Package	Application	Chapter	Authors
nimble	Reversible jump MCMC	2	de Valpine <i>et al.</i> , (2017)
	Laplace density function (double exponential)	5	
ElemStatLearn	Dataset: South African Heart Disease	3	Hastie, Tibshirani & Friedman (2019)
	Dataset: Prostate	5	
rstanarm	Bayesian Logistic regression	3	Goodrich <i>et al.</i> , (2020)
bayesplot	Bayesian Logistic Regression: posterior density plots	3	Gabry & Mahr (2020)
tm	Naïve Bayes: text mining	3	Feinerer & Hornik (2019)
e1071	Naïve Bayes	3	Meyer <i>et al.</i> , (2019)
	Support vector classifiers and support vector machines	5	
BoomSpikeSlab	Spike-and-slab priors	5	Scott (2020)
BayesVarSel	Zellner's $g$ -Prior	5	Garcia-Donato & Forte (2018)
monomvn	Bayesian Lasso and Ridge Regression	5	Gramacy, Moler & Turlach (2019)
bnlearn	Bayesian belief networks	6	Scutari (2010)
depmixS4	Hidden Markov Models	6	Visser & Speekenbrink (2010)
dlm	Dynamic linear models: Kalman Filter	6	Petris (2010)
mrf2d	Markov Random Fields	6	Freguglia & Garcia (2020)
crfsuite	Conditional Random Fields	6	Wijffels & Okazaki (2018)

## 1.5 CHAPTER OUTLINES

There are 5 main chapters in the body of this thesis, each focusing on a broad class of Bayesian machine learning methods. Chapter 2 begins with an overview of the notation used throughout the thesis as well as a summary of the main probability theory concepts that feature heavily in later chapters. The remainder of Chapter 2 provides an outline of Monte Carlo methods since these are used in many applications of machine learning for both computational and analytical tractability reasons.

Chapters 3 and 4 cover Bayesian approaches to classification and regression problems, respectively. Specifically, Bayesian logistic regression and Bayesian naïve Bayes are considered in Chapter 3 along with an explanation of Laplace approximation. Chapter 4 provides a detailed description of a fully Bayesian approach to linear regression. Further, a detailed description of the expectation-maximization algorithm, used in several places in this thesis, is also given in Chapter 4.

Chapter 5 provides a description and comparison of four different Bayesian variable selection techniques. Also, although not strictly a variable selection technique but rather a sparse Bayesian learning method, Chapter 5 includes a discussion and derivation of relevance vector machines and their use in both regression and classification problems. Finally, Chapter 6 considers probabilistic graphical models and the common scenarios in which they are applied. This chapter covers various directed and undirected graphical models and has a greater application focus in contrast to the other chapters. The schematic given in Figure 1.2 below provides a visual overview of the chapter outlines.



**Figure 1.2: Schematic overview of the various topics covered.**

## CHAPTER 2

### PROBABILITY THEORY AND MONTE CARLO METHODS

#### 2.1 PROBABILITY THEORY

Probability theory forms the backbone for many machine learning techniques. The first part of this chapter highlights some of the most prominent aspects of probability theory that will be used throughout this thesis, assuming that the basics (such as the axioms of probability, the law of conditional probability, the law of total probability etc.) are already well known. Most machine learning textbooks begin with an overview of introductory statistics and hence a convenient summary of these basic probability concepts can be found in Theodoridis (2015: 10–17) or Bishop (2006: 12–32). Also included in this chapter is a summary of the multinomial and Dirichlet distribution since these two distributions feature several times throughout the thesis and are not always as familiar as other distributions. Derivations proving conjugacy between any relevant distributions are also shown.

Throughout this thesis, consistent notation will be used. Where there is an overlap in notation or any possible ambiguity, it will be made clear from the context what the relevant definition is. The following table provides the relevant symbols that will be used.

**Table 2.1: Summary of notation.**

Symbol	Definition
$\mathbf{x}: n \times 1$	A column vector
$\mathbf{x}^T: 1 \times n$	A row vector
$\mathbf{X}: n \times p$	A matrix with $n$ rows and $p$ columns
$X$	A univariate random variable
$\mathcal{D}$	The dataset of interest
$P(\cdot)$	Probability
$p(\cdot)$	Probability density function

For most of the problems in the chapters to follow, there will typically be an observed dataset,  $\mathcal{D}$ , on which we will apply the various Bayesian machine learning techniques. It will further be assumed that the dataset contains a total of  $N$  realizations (or  $T$  when working with Markov chains) which arise from some underlying, and unknown, probability distribution.

### 2.1.1 Bayes' Theorem

Bayes' theorem takes its name from the English mathematician Thomas Bayes (1702-1761) who was the first person to propose and develop the initial foundations of the theory. However, it was only after further research and development by the French mathematician Pierre-Simon Laplace (1749-1827) that Bayes' theorem became popular and a greater use of Bayesian modelling arose (Theodoridis, 2015:84).

As mentioned in the introductory chapter, Bayes' theorem forms the foundation for all Bayesian modelling approaches. We consider the training dataset  $\mathcal{D}$  and suppose that we are interested in some variable  $\theta$  based on this observed data. The quantity of interest is thus  $p(\theta|\mathcal{D})$ , and, using Bayes Theorem, we express this as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_{\theta} p(\mathcal{D}|\theta)p(\theta)d\theta}. \quad (2.1)$$

Expressing this in words, the posterior distribution, i.e. the distribution of  $\theta$  given the observed data, is equal to the likelihood function of the model,  $p(\mathcal{D}|\theta)$ , multiplied by some prior belief about the parameter set,  $p(\theta)$ , divided by the normalizing constant  $p(\mathcal{D})$ . Since this normalizing constant does not depend on  $\theta$  and hence remains constant over all model parameters, it is often excluded, and the posterior distribution is represented using proportionality. Thus, the posterior distribution is commonly expressed as

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) \quad (2.2)$$

which translates in words to

$$\text{posterior} = \text{likelihood} \times \text{prior}. \quad (2.3)$$

The likelihood function of a dataset provides an indication of how well the sample data fits a chosen model for given values of the parameters. This likelihood function, multiplied by the prior distribution, results in a posterior distribution for  $\theta$  that incorporates information obtained directly from the observed data as well as prior beliefs of what values the parameters are most likely to assume. The normalizing constant is commonly referred to as the model evidence and plays a crucial role in Bayesian statistics. This quantity is typically difficult to compute analytically and so approximation methods are often required to estimate it. More attention will be given to the model evidence in Section 4.3.

Bayes theorem allows for the posterior distribution of the model parameters to be obtained, but it is not always this posterior distribution that is of interest. In practice, it is often of greater interest to be able to predict values of the response value when new data arises. Thus, the predictive distribution, denoted by  $p(y|x, \mathcal{D})$ , provides the distribution of the response variable based on the observed training data  $\mathcal{D}$  and the new data  $x$ . Examples of the predictive distribution will be given



in Section 3.2.2 for the logistic regression model, Section 4.2.3 for the multiple linear regression model and Section 5.6.2 for relevance vector regression.

### **2.1.2 The Prior Distribution**

The posterior distribution can be regarded as a trade-off between the likelihood of the data and the prior distribution. This means that different prior distributions will naturally have a large influence on what the resulting form of the posterior distribution will be, especially in smaller datasets where the likelihood function is less dominating. The choice of what prior to use in a model is a contentious topic in Bayesian statistics due to their ability to have such a large influence on the resulting posterior distribution. Prior distributions are typically either considered to be informative or non-informative depending on the relative amount of information that they carry. Section 4.2.1 provides further insight into the relationship and effects of the likelihood and prior on the posterior.

Naturally, an informative prior is one which expresses specific beliefs about the values of the model parameters. For example, if we were considering modelling the distribution of the outcomes of flipping a coin multiple times, it would be natural to assume that the probability of obtaining a head or tail would be equal, i.e. 50%, since a common coin is typically unbiased. Thus, we would most likely use a prior distribution with a large concentration of values around 0.5. In this scenario, we are incorporating prior knowledge about the event into our model in addition to considering the likelihood of the data. The resulting posterior distribution will then take a shape that is a combination of both the likelihood, and this chosen prior.

On the other hand, if we were performing an analysis but had no prior knowledge regarding the parameters of the model, or wanted the likelihood function of the data to carry more weight in the analysis, we would use a non-informative prior. Non-informative priors – as their names suggests – should (ideally) provide no new specific information about the model parameters. These priors are specifically chosen so that their influence on the posterior will be as small as possible, in other words, “letting the data speak for themselves” (Bishop, 2006:118). Hence, the idea behind the use of such priors is to derive a posterior distribution that is influenced solely by the likelihood function of the data. However, not all non-informative priors will always be truly non-informative for all distributions and so caution must be given to selection of non-informative prior.

### **2.1.3 The Multinomial Distribution**

This thesis works extensively with the mathematical manipulation of several probability distributions, most of which are commonly encountered in the literature and are usually very familiar. Less commonly encountered distributions include the multinomial and Dirichlet distributions. Since these distributions will be work with extensively in the chapters to follow, they will now be discussed in turn.

The multinomial distribution can be regarded as a generalization of the binomial distribution. The distribution of a random variable that can take on one of two possible outcomes with a probability of  $p$  and  $q = 1 - p$  can be modelled by the binomial distribution. If we extend this to the case where there are more than 2 possible outcomes, we can model the distribution using the multinomial distribution.

We consider the discrete random variable  $X$  and suppose that there are a total of  $S$  possible outcomes for  $X$  where each of these outcomes occurs with a probability of  $\mu_s, s = 1, \dots, S$ . This means that  $\mu_s$  represents the probability that  $X = s$  with  $\sum_{s=1}^S \mu_s = 1$ . After  $N$  outcomes have been observed, the probability of  $X_1 = x_1, \dots, X_S = x_S$  can be modelled by the multinomial distribution, which is given by

$$p(\mathbf{x}; \boldsymbol{\mu}) = \binom{N}{x_1 \dots x_S} \prod_{s=1}^S \mu_s^{x_s} \quad (2.4)$$

with  $\mathbf{x} = [x_1, \dots, x_S]^T, \boldsymbol{\mu} = [\mu_1, \dots, \mu_S]^T$  and where  $x_s$  denotes the number of times that state  $s$  occurred so that  $\sum_{s=1}^S x_s = N$  (Murphy, 2012:35).

We can alternatively express the multinomial coefficient in the following equivalent form which will be useful later,

$$\binom{N}{x_1 \dots x_S} = \frac{N!}{x_1! \dots x_S!} = \frac{\Gamma(N+1)}{\prod_{s=1}^S \Gamma(x_s+1)} \quad (2.5)$$

where the Gamma function is given by  $\Gamma(n+1) = n\Gamma(n)$  and  $\Gamma(1) = 1$ .

The log-likelihood function for the multinomial distribution is given by

$$\begin{aligned} \ell &= \log N! - \sum_{s=1}^S \log x_s! + \sum_{s=1}^S x_s \log \mu_s \\ &\propto \sum_{s=1}^S x_s \log \mu_s. \end{aligned} \quad (2.6)$$

In order to determine the maximum likelihood estimate for each  $\mu_s, s = 1, \dots, S$ , we require the use of Lagrange multipliers to incorporate the constraints into the estimation. Hence, we aim to maximize

$$\sum_{s=1}^S x_s \log \mu_s + \lambda \left( \sum_{s=1}^S \mu_s - 1 \right) \quad (2.7)$$

with respect to  $\mu_s$ . Taking the derivative of equation 2.7 with respect to  $\mu_s$  and setting the result equal to zero gives:

$$\begin{aligned}
 \frac{\partial}{\partial \mu_s} \left\{ \sum_{s=1}^S x_s \log \mu_s + \lambda \left( \sum_{s=1}^S \mu_s - 1 \right) \right\} &= 0 \\
 \Rightarrow \frac{x_s}{\mu_s} + \lambda(1) &= 1 \\
 \Rightarrow \mu_s &= -\frac{x_s}{\lambda}.
 \end{aligned} \tag{2.8}$$

Substituting this result back into the constraint, gives,

$$\begin{aligned}
 \sum_{s=1}^S \mu_s &= 1 \\
 \Rightarrow \sum_{s=1}^S \left( -\frac{x_s}{\lambda} \right) &= 1 \\
 \Rightarrow -\sum_{s=1}^S x_s &= \lambda \\
 \Rightarrow -N &= \lambda.
 \end{aligned} \tag{2.9}$$

Hence, the maximum likelihood estimator for  $\mu_s$  is given by

$$\hat{\mu}_s = \frac{x_s}{N}, \quad s = 1, \dots, S \tag{2.10}$$

which is simply the fraction of times that the  $s$ th outcome occurred, which makes intuitive sense.

One special case of the multinomial distribution occurs when  $N = 1$ . This means that the vector  $x$  will be a vector of zeros with only one component equal to one corresponding to the respective outcome that was observed. For example, if we have a total of  $S = 5$  possible states and we observe state 3 as the outcome of the event, the variable  $x$  will be encoded as  $x = [0, 0, 1, 0, 0]^T$ . This special case defines the categorical distribution, and the density function for the categorical distribution is given by

$$p(x; \mu) = \prod_{s=1}^S \mu_s^{I(x_s=1)} \tag{2.11}$$

where again the constraint  $\sum_{s=1}^S \mu_s = 1$  applies. This distribution is also sometimes referred to as the Multinoulli distribution. This name comes from the fact that this distribution is simply a generalization of the Bernoulli distribution which has been extended to model a discrete variable which can take on more than two possible outcomes. Alternatively, the multinomial distribution can also be regarded as the distribution resulting from  $N$  independent trials of the categorical distribution.

### 2.1.4 The Dirichlet Distribution

The Dirichlet distribution can be regarded as the multivariate generalization of the Beta distribution. The density function for a variable  $\boldsymbol{\mu} = [\mu_1, \dots, \mu_S]^T$  that follows a Dirichlet distribution is given by

$$p(\boldsymbol{\mu}; \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{s=1}^S \mu_s^{\alpha_s-1} \quad (2.12)$$

where

$$B(\boldsymbol{\alpha}) = \frac{\prod_{s=1}^S \Gamma(\alpha_s)}{\Gamma(\sum_{s=1}^S \alpha_s)} \quad (2.13)$$

and the parameters  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_S]^T$  satisfy  $\sum_s \alpha_s = 1$ .

The Dirichlet distribution is often used as a prior distribution in Bayesian learning, as will be the case in Chapter 3, since it is conjugate to the multinomial and categorical distributions. A prior distribution is said to be a conjugate prior if the posterior distribution belongs to the same family of distributions as the prior. This is a convenient result which simplifies the mathematics of an analysis by obtaining a posterior distribution that exists in closed form and follows a known density function. Hence, to prove conjugacy, the resulting posterior distribution needs to be shown to fall into the same family of distributions as the prior distribution.

If we consider the Dirichlet prior and the multinomial likelihood, the posterior is given by

$$\begin{aligned} p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\alpha}) &\propto p(\mathcal{D}|\boldsymbol{\mu})p(\boldsymbol{\mu}|\boldsymbol{\alpha}) \\ &\propto \left\{ \prod_{s=1}^S \mu_s^{x_s} \right\} \times \left\{ \prod_{s=1}^S \mu_s^{\alpha_s-1} \right\} \\ &= \prod_{s=1}^S \mu_s^{x_s+\alpha_s-1} \end{aligned} \quad (2.14)$$

which is the kernel for a Dirichlet distribution with parameters  $\boldsymbol{\alpha} + \boldsymbol{x}$ . This proves that the Dirichlet distribution is conjugate to the multinomial distribution, as mentioned. Clearly, since we saw that the categorical distribution was a special instance of the multinomial distribution, it follows immediately that the Dirichlet distribution will also be conjugate to the categorical distribution (Bishop, 2006:77).

## 2.2 MONTE CARLO METHODS

In many applications of Bayesian learning, it is often analytically intractable, or computationally infeasible, to evaluate a quantity exactly. As a result, it is common to use approximation

techniques to obtain estimates of the quantity of interest. Typically, these methods are used to estimate the value of an integral, or to sample from a posterior distribution, instead of determining the value of estimates exactly or finding closed-form solutions for certain integrals.

Monte Carlo methods encompass a wide class of sampling techniques that are particularly prominent in the Bayesian world. Since Bayesian methods often involves working with mixtures of various distributions, determining the posterior distribution in a closed form is more often than not impossible. Although sampling techniques take up a wide space in Bayesian literature, only those methods which are used directly in this paper or are typically used with any of the described techniques in the following chapters, will be presented.

### 2.2.1 Markov Chains

In order to define a Markov chain, we consider the situation when the data arise sequentially due to some form of stochastic process. The basic idea behind a Markov chain is to assume that to determine the next observation in the sequence, we require only the knowledge of the present observation. In other words, the future can be modelled independently of the past given only the current knowledge. This means that we can make the assumption

$$p(x_t | x_1, \dots, x_{t-1}) = p(x_t | x_{t-1}). \quad (2.15)$$

Hence, a Markov model, or Markov chain, arises when the joint distribution function of a set of data observations is modelled in the following way

$$\begin{aligned} p(x_1, x_2, \dots, x_T) &= p(x_1)p(x_2|x_1)p(x_3|x_2, x_1) \dots p(x_T|x_{T-1}, x_{T-2}, \dots, x_2, x_1) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_2) \dots p(x_T|x_{T-1}) \\ &= p(x_1) \prod_{t=2}^T p(x_t|x_{t-1}) \end{aligned} \quad (2.16)$$

assuming that we are working with discrete time steps (Murphy, 2012:589). Strictly speaking, this would be referred to as a first-order Markov chain. We could similarly obtain higher-order Markov chains by assuming that a future observation is dependent only on the previous two observations, that is,

$$p(x_t | x_1, \dots, x_{t-1}) = p(x_t | x_{t-1}, x_{t-2}). \quad (2.17)$$

Clearly, a second-order Markov chain will do better in capturing long-range correlations than a first-order Markov chain, but it is still quite a restrictive assumption that requires more parameters to estimate. Building even higher order Markov chains becomes infeasible since the size of the parameter set quickly increases to an impractical amount.

A Markov chain is fully defined by a matrix of transition probabilities,  $\mathbf{A}$ , and an initial distribution given by a vector of initial state probabilities. This vector of initial probabilities will be denoted by

$\mathbf{p}^{(0)}$  and the  $ij$ th element of the transition matrix represents the conditional probability of moving to state  $j$  given that you are currently in state  $i$ , that is, at time point  $t$ ,

$$\mathbf{A}_{ij}^{(t)} = p(x_{t+1} = j | x_t = i) \quad (2.18)$$

where the elements of the transition matrix satisfy

$$0 \leq \mathbf{A}_{ij} \leq 1 \quad \text{with} \quad \sum_j \mathbf{A}_{ij} = 1. \quad (2.19)$$

A Markov chain is said to be homogeneous, or stationary, if these transition probabilities are equal for every given time point  $t$  (Theodoridis, 2015: 721), that is,

$$\mathbf{A}_{ij}^{(t)} = \mathbf{A}_{ij}^{(s)}, \quad \forall s \neq t; s, t = 1, \dots, T. \quad (2.20)$$

Hence, if we have a homogeneous Markov chain, then the vector representing the probability of being in each state after  $T$  time steps can be expressed as

$$\mathbf{p}^{(T)} = \mathbf{A}\mathbf{p}^{(T-1)} = \mathbf{A}\mathbf{A}\mathbf{p}^{(T-2)} = \dots = \mathbf{A}^T\mathbf{p}^{(0)}. \quad (2.21)$$

The basis of Markov chain Monte Carlo methods discussed in the next section relies on the Markov chain being of a very specific type such that after many time steps the resulting vector of state probabilities will converge, regardless of the initial probability vector. In other words, a Markov chain that satisfies,

$$\lim_{T \rightarrow \infty} \mathbf{p}^{(T)} = \lim_{T \rightarrow \infty} \mathbf{A}^{(T)}\mathbf{p}^{(0)} \quad (2.22)$$

is known as an ergodic Markov chain and this limit will be independent of the choice of  $\mathbf{p}^{(0)}$  (Theodoridis, 2015: 723). Further, this stationary distribution to which it converges will be unique. Hence, sampling methods that make use of the theory of Markov chains aim to obtain this stationary distribution by taking a large enough number of samples to be sure that the chain converges to the desired distribution.

### 2.2.2 Markov Chain Monte Carlo

Often in Bayesian inference we find ourselves faced with the task of evaluating integrals of the form

$$\int g(\boldsymbol{\theta}) f_{\boldsymbol{\theta}|x}(\boldsymbol{\theta}) d\boldsymbol{\theta} \quad (2.23)$$

where the subscript in  $f_{\boldsymbol{\theta}|x}(\boldsymbol{\theta})$  refers to the posterior distribution, i.e. the distribution of the parameter set  $\boldsymbol{\theta}$  after the data have been observed.

Markov Chain Monte Carlo (MCMC) methods encompass a range of different algorithms that can obtain estimates of such integrals. Two popular methods that will be looked at in this section are the Metropolis-Hastings algorithm and Gibbs sampling.

In many situations, the aim is to obtain samples from some target distribution  $p(\mathbf{z})$ , however, it is often difficult to sample directly from  $p(\mathbf{z})$  when  $p(\mathbf{z})$  does not take the form of any commonly encountered density function. However, as is typically the case, we will assume that for a given value of  $\mathbf{z}$ , it is not difficult to evaluate the function at the point  $\mathbf{z}$ , excluding the normalizing constant. Thus, denoting the normalizing constant for the distribution by  $Z$ , we can express the target function as

$$p(\mathbf{z}) = \frac{1}{Z} \tilde{p}(\mathbf{z}) \quad (2.24)$$

and will throughout suppose that we are easily able to evaluate  $\tilde{p}(\mathbf{z})$ .

In many sampling algorithms, we make use of a proposal distribution from which sampling is not too difficult to do. This proposal distribution must fully envelope the target distribution (i.e. the target distribution must be less than or equal to the proposal distribution at all points). The general idea in MCMC methods is to generate a sample from the proposal distribution and then either reject or accept this sample based on a specific evaluation criterion.

### 2.2.2.1 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm (Hastings, 1970) evolves by starting with some proposal distribution, denoted by  $q(\mathbf{z})$ , from which it is easy to sample. Then, at the  $m$ th step of the algorithm, we generate a sample  $\mathbf{z}_m$  from the proposal distribution, keep a record of this sample, and adapt the proposal distribution to become dependent on this sample. The form of the proposal distribution at the  $m$ th step of the algorithm is then given by  $q(\mathbf{z}|\mathbf{z}_m)$ . The next sample for use in the algorithm is then generated from this proposal distribution so that the sequence of samples  $\mathbf{z}_1, \dots, \mathbf{z}_m$  forms a Markov chain. This set of samples will clearly be highly correlated, hence this cannot be regarded as an independent sample (Bishop, 2006:583). However, an independent sample can easily be obtained from this by selecting out every  $k$ th value in the sequence – assuming that the sequence generated is long enough.

We consider now the criterion for accepting the generated sample point. At the  $m$ th step of the algorithm, we obtain a sample from the proposal distribution,  $q(\mathbf{z}|\mathbf{z}_m)$ , and we will denote this sample by  $\mathbf{z}^*$ . The probability of accepting this sample point is then given by the following expression,

$$\alpha(\mathbf{z}^*, \mathbf{z}_m) = \min \left\{ 1, \frac{\tilde{p}(\mathbf{z}^*)q(\mathbf{z}_m|\mathbf{z}^*)}{\tilde{p}(\mathbf{z}_m)q(\mathbf{z}^*|\mathbf{z}_m)} \right\}. \quad (2.25)$$

We can see from this acceptance probability why it does not matter if we are unable to determine the value of the normalizing constant for  $p(\mathbf{z})$  since this constant will cancel out in the fraction anyway. One special case of this algorithm, which was in fact proposed before the Metropolis-Hastings algorithm, is known simply as the Metropolis algorithm (Metropolis *et al.*, 1953) which works only with symmetric proposal distributions, so that s

$$q(\mathbf{z}_m|\mathbf{z}^*) = q(\mathbf{z}^*|\mathbf{z}_m) \quad (2.26)$$

which means that the acceptance probability criterion reduces to

$$\alpha(\mathbf{z}^*, \mathbf{z}_m) = \min \left\{ 1, \frac{\tilde{p}(\mathbf{z}^*)}{\tilde{p}(\mathbf{z}_m)} \right\}. \quad (2.27)$$

Finally, to determine whether the sample  $\mathbf{z}^*$  will be accepted, a random number  $u$  is generated from a uniform distribution over the interval  $(0,1)$  and if  $\alpha(\mathbf{z}^*, \mathbf{z}_m) > u$ , the sample  $\mathbf{z}^*$  will be accepted. If accepted, we set  $\mathbf{z}^{m+1} = \mathbf{z}^*$  and the process repeats itself. If the sample is not accepted, we set  $\mathbf{z}^{m+1} = \mathbf{z}^m$  and another sample from  $q(\mathbf{z}|\mathbf{z}^{m+1}) = q(\mathbf{z}|\mathbf{z}^m)$  is drawn. Typically, a burn-in period is used whereby the first few values generated from the algorithm are discarded. This is to ensure that the sample obtained has converged.

We consider an illustrative example of implementing the Metropolis-Hastings algorithm for the situation where the data are generated from a univariate Gaussian distribution with a known precision parameter  $\tau$  and unknown mean parameter  $\mu$ . Precision is simply equal to the inverse of the variance, i.e.  $\tau = \frac{1}{\sigma^2}$ , where  $\sigma^2$  denotes the variance for the Gaussian distribution. Precision is commonly worked with instead of the variance in a Bayesian context. Gaussian distributions are conjugate to themselves and so we assign a Gaussian prior to the mean parameter, such that,

$$p(\mu) \sim \mathcal{N}(\mu_0, \tau_0). \quad (2.28)$$

In Section 4.2, the posterior distribution for this scenario will be derived in full. That result will simply be used here directly. Hence, the posterior distribution for the mean is given by

$$p(\mu|y) \sim \mathcal{N}(\mu_N, \tau_N) \quad (2.29)$$

where

$$\mu_N = \frac{\tau_0 \mu_0 + \tau \sum_{i=1}^N x_i}{\tau_0 + N\tau} \quad (2.30)$$

and

$$\tau_N = \tau_0 + N\tau. \quad (2.31)$$



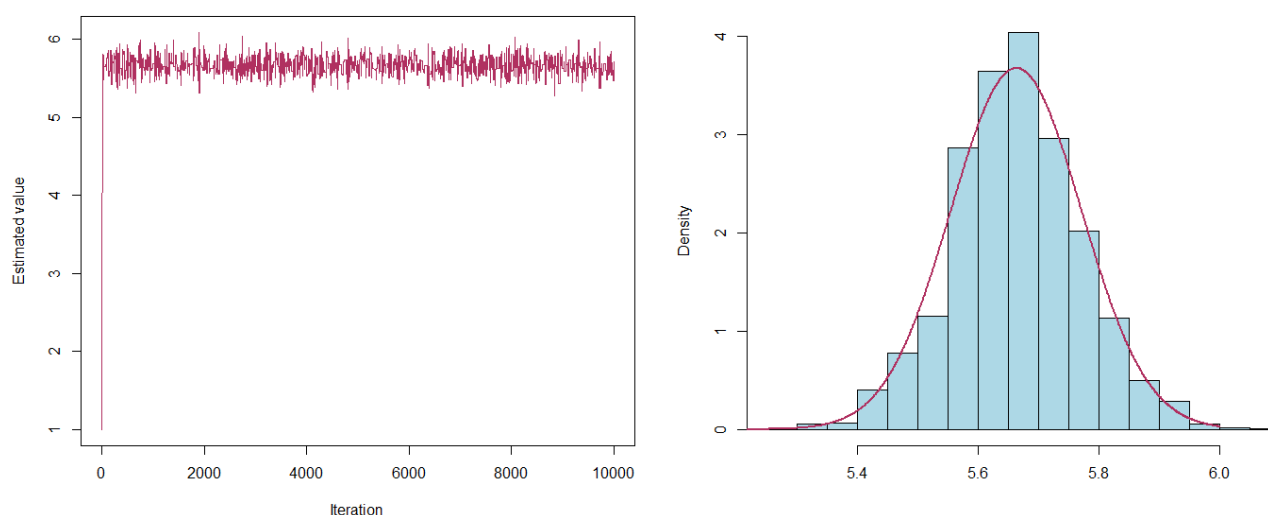
We will use the Metropolis-Hastings algorithm to sample from the posterior distribution of  $\mu$  using values of  $\tau = 4$ ,  $\mu_0 = 0$  and  $\tau_0 = 5$ . The true value of  $\mu$  used to generate the data was  $\mu = 6$ . The proposal distribution is also Gaussian given by

$$\mu^{(m+1)} \sim \mathcal{N}(\mu^{(m)}, 1) \quad (2.32)$$

and a starting value of  $\mu^{(0)} = 1$  is used.

As mentioned, an important aspect with MCMC algorithms is that, as the number of samples increases, the distribution obtained from the sampled points will converge to a fixed distribution regardless of the initial starting position. The distribution to which it converges is known as the equilibrium, or stationary, distribution and this property of converging to a stationary distribution is known as ergodicity (Bishop, 2006: 540). The sample mean and sample standard deviation from the stationary distribution can then be used as estimates for the corresponding population parameters. To ensure that this convergence occurs, a burn-in period is typically used whereby a certain amount of the first few samples obtained are discarded and the distribution is approximated using the remaining, non-discarded, values. Convergence to the stationary distribution can typically be seen through inspection of the trace plot (i.e. a plot of the sampled value at each iteration) such as the one given on the left in Figure 2.1.

The results of the Metropolis-Hastings algorithm applied to the Gaussian problem are given below.



**Figure 2.1: Metropolis-Hastings output for the Gaussian simulation example.**

The trace plot on the left in Figure 2.1 shows that the algorithm converged quickly to the posterior distribution of  $\mu$ . The histogram on the right in Figure 2.1 shows the estimated posterior distribution obtained from the algorithm and the red curve displays the true posterior distribution. This true posterior is known and exists in closed form since a conjugate prior was used. The burn-in period

used was 1000 iterations. The true parameter values of the posterior distribution are  $\mu_N = 5.6634$  and  $\tau_N = 85$ . The estimated parameter values obtained from the algorithm are  $\mu_N^* = 5.6640$  and  $\tau_N^* = 86.82$  which are clearly very close to their true values.

### 2.2.2.2 Gibbs Sampling

Gibbs sampling, proposed by brothers Geman & Geman (1984) can be regarded as a specific instance of the Metropolis algorithm when all of the conditional distributions are known. The target distribution is  $p(z_1, \dots, z_M)$  and Gibbs sampling works by individually replacing each of the variables in the distribution by a sample generated from the conditional distribution of the remaining variables. Hence, beginning with some initial state of the algorithm, at the  $m$ th step, we generate

$$z_j^{m+1} \sim p(z_j | z_1^{m+1}, \dots, z_{j-1}^{m+1}, z_{j+1}^m, \dots, z_M^m) \quad (2.33)$$

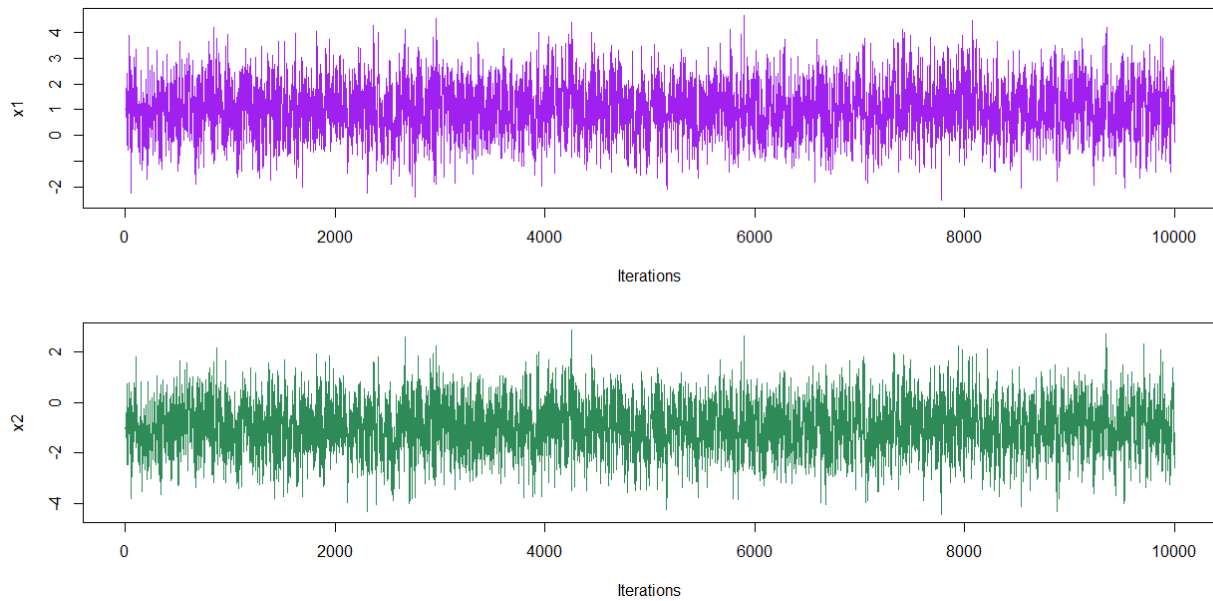
for all  $j = 1, \dots, M$ . Thus, we notice that we use the updated version for all previously sampled variables in the conditional distribution for the  $j$ th variable. This is a specific instance of the Metropolis-Hastings algorithm where all steps are accepted since all acceptance probabilities will be equal to one.

A simple example demonstrating how the Gibbs sampler works is to consider the situation where we wish to obtain samples from a bivariate normal distribution (Rizzo, 2007: 263-265). Consider  $\mathbf{x} = [x_1, x_2]^T$  distributed according to a bivariate normal distribution with mean vector  $\boldsymbol{\mu} = [\mu_1, \mu_2]^T$ , covariance matrix  $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$  and correlation  $\rho$ . It is well known that each of the marginal distributions is also normally distributed, i.e.  $x_i \sim \mathcal{N}(\mu_i, \sigma_i^2), i = 1, 2$ . Gibbs sampling requires the specification of the conditional distributions. The conditional distributions for a bivariate normal distribution, expressed in terms of the correlation  $\rho$ , are given by

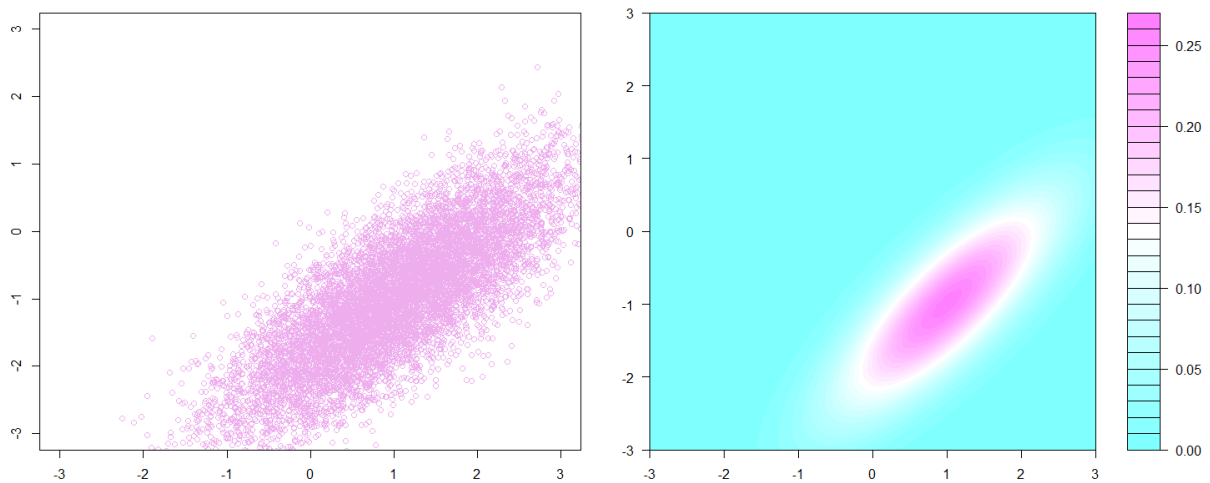
$$p(x_1 | x_2) \sim \mathcal{N}\left(\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(x_2 - \mu_2), (1 - \rho^2)\sigma_1^2\right) \quad (2.34)$$

$$p(x_2 | x_1) \sim \mathcal{N}\left(\mu_2 + \frac{\rho\sigma_2}{\sigma_1}(x_1 - \mu_1), (1 - \rho^2)\sigma_2^2\right). \quad (2.35)$$

The results of the Gibbs sampler are provided below using values of  $\mu_1 = 1, \mu_2 = -1, \sigma_1^2 = \sigma_2^2 = 1$  and  $\rho = 0.8$ . Figure 2.2 shows the trace plots for  $x_1$  and  $x_2$  and Figure 2.3 shows the simulated density function in the left-hand plot and the true bivariate normal density function in the right-hand plot. By inspection of Figure 2.3, the Gibbs sampling algorithm can be seen to have done well to generate samples from the bivariate normal distribution.



**Figure 2.2: Trace plots for each variable in the bivariate normal model.**



**Figure 2.3: Simulated data using the estimates obtained from the Gibbs sampling algorithm (left) and the true bivariate density function (right).**

### 2.2.2.3 Reversible Jump MCMC

We now extend the Metropolis-Hastings algorithm to allow for state-spaces which have varying dimensions so that the Markov chain generated is able to move between such models. This is known as reversible jump Markov chain Monte Carlo (Green, 1995). This extension will be applied in Chapter 5 to the Bayesian LASSO and ridge regression variable selection problem. Reversible jump MCMC is often applied in variable selection problems where there are competing models,  $\mathcal{M}_k, k = 1, \dots, K$ , of varying dimensionality due to the inclusion and exclusion of model parameters.

Here, the proposal distribution and target distributions now have density functions defined over spaces with possibly different dimensions. Another common application of reversal jump MCMC is to sample from a mixture of Gaussian densities where the number of models in the mixture is unknown.

For the case of variable selection, the target density function simply becomes the posterior distribution for the model parameters for a given model choice. We denote by  $p(\boldsymbol{\theta}|\mathcal{D})$  and  $p(\boldsymbol{\theta}'|\mathcal{D})$  the posterior distributions for models  $\mathcal{M}$  and  $\mathcal{M}'$ , respectively. In the same manner as the Metropolis-Hastings algorithm, we transition from the current parameter state  $\boldsymbol{\theta}$  based on model  $\mathcal{M}$  by obtaining a sample  $\boldsymbol{\theta}'$  from model  $\mathcal{M}'$  from the proposal distribution  $q(\boldsymbol{\theta}, \boldsymbol{\theta}')$ . The acceptance probability for this sample is then given by

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min \left\{ 1, \frac{p(\boldsymbol{\theta}|\mathcal{D})q(\boldsymbol{\theta}, \boldsymbol{\theta}')}{p(\boldsymbol{\theta}'|\mathcal{D})q(\boldsymbol{\theta}', \boldsymbol{\theta})} \right\}. \quad (2.36)$$

Hence, starting at some initial state, the algorithm evolves by first updating the model parameters for a fixed model, followed by making a step to a new model (of possibly different dimensionality) using the acceptance probability given in Equation 2.36. Clearly, the models for which the data shows a greater favour towards will have higher acceptance probabilities and so more samples will be simulated from these models. This is the general intuition underlying reversible jump MCMC. The full theory behind the reversible jump MCMC algorithm relies heavily on the use of measure theory which is beyond the scope of this thesis and has hence been omitted.

Instead, to practically illustrate the method, we consider the simple implementation of the algorithm in the multiple linear regression scenario. We consider the multiple linear regression model

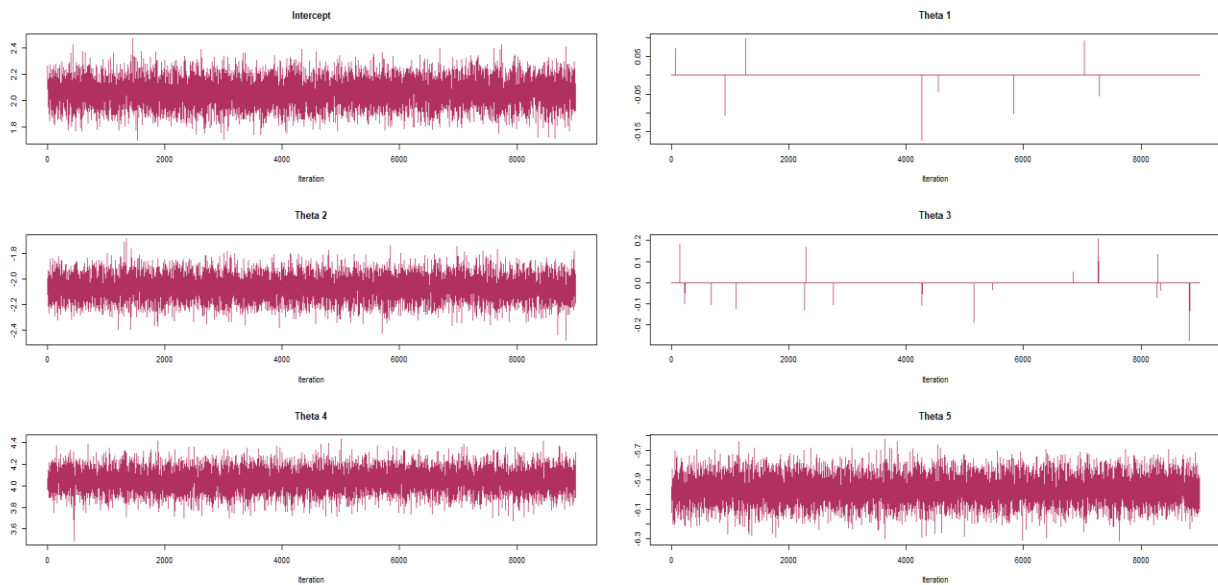
$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5 + \varepsilon \quad (2.37)$$

where  $\varepsilon$  denotes the error term and the true value of the parameter set is given by

$$\boldsymbol{\theta} = [2, 0, -2, 0, 4, -6]^T. \quad (2.38)$$

Hence,  $x_1$  and  $x_3$  should not be included in the model. A total of  $N = 100$  points were generated from this model and a zero-mean, Gaussian random noise term with a variance of 1 was added to each point. Given this observed data, reversible jump MCMC can be used for variable selection whereby the algorithm should hopefully return a model excluding variables  $x_1$  and  $x_3$  since these are not in the true model by construction. The reversible jump MCMC algorithm works by moving between the total possible combinations of  $2^6 = 64$  possible models obtained from including and excluding each of the six variables in the model. Hence, at each iteration, the algorithm will update the parameters in the current model using the normal Metropolis-Hastings algorithm to perform the update, and then take a step towards a newly proposed model using the acceptance

probability given in Equation 2.36. The trace plots obtained from this algorithm are given in Figure 2.4.



**Figure 2.4: Trace plots obtained from reversible jump MCMC.**

From Figure 2.4, since the true model did not actually contain  $x_1$  and  $x_3$ , the algorithm very seldom visited those models since the acceptance probability would have been very small. This can be seen by the very few samples that were non-zero in the trace plots. In other words, when the newly proposed model and parameter values were further from the true values of the model, the data would show less favour towards this model and so that step to the new model will most likely be rejected. Therefore, the posterior samples resulted in very few non-zero simulated parameter values for  $x_1$  and  $x_3$  – the data would not be likely to accept a step towards models containing those parameters.

The remaining variables in the model have trace plots all oscillating close to the true values of the parameters since the models containing those sets of parameters would have been accepted with a high probability in each step. Hence, the reversible jump MCMC algorithm is a useful tool for simulating values from target distributions where there are possibly varying dimensionalities and can thus also be used as a method of performing variable selection.

## 2.3 APPROXIMATE BAYESIAN COMPUTATION

MCMC methods are useful and powerful techniques for approximating distributions that are difficult, or impossible, to evaluate numerically. However, since they are reliant on large numbers of iterations, they can also become impractical if the dataset is very large or the function to approximate is very complex. This is becoming more and more of an issue as the availability of data is rapidly increasing and resulting in significantly large datasets. Furthermore, the focus is

often put on estimation of the posterior distribution, but this assumes that we are easily able to determine the likelihood function in closed form. As data is expanding, this is becoming increasingly uncommon. One alternative is to simplify the chosen model so that the likelihood function becomes straightforward to evaluate, but analytical tractability for the sake of computational and aesthetical ease does not always result in the best outcomes. A well-suited approximation can often outperform a model which has been simplified to exist in a closed form.

This issue prompted the introduction of approximate Bayesian computation (ABC) which was first proposed by Rubin (1984), but only termed as such by Beaumont *et al.*, (2002). The general intuition behind the approach is to simulate values from a prior distribution and accept only those samples for which the difference between the likelihood function (obtained from each sample) and the observed values are smaller than some pre-specified amount. The ABC rejection sampling algorithm was formalized by Tavaré *et al.*, (1997) and is summarized below for a given parameter set  $\theta$  and a pre-selected prior distribution  $p(\theta)$ . The algorithm assumes that it is possible to simulate values from the likelihood function for given values of the parameter set.

#### **Algorithm: ABC rejection sampling**

Suppose that we are given a sample  $y$  for which the likelihood function,  $p(y|\theta)$ , can be sampled from for given values of the parameter set  $\theta$ . Select a distance metric, denoted by  $\|\cdot\|$  and an approximation threshold,  $\epsilon > 0$ .

1. Simulate a set of parameter values from the prior distribution

$$\theta^* \sim p(\theta)$$

2. Using these sampled parameter values, simulate values from the likelihood function

$$y^* \sim p(y|\theta^*)$$

3. Accept the proposed sample from the prior distribution,  $\theta^*$ , if

$$\|y^* - y\| < \epsilon$$

otherwise reject this sample. Return to step 1 and repeat.

Clearly, if we make use of this approach, the set of all accepted  $\theta^*$  can be regarded as samples from the posterior distribution. Hence, the distribution of the accepted  $\theta^*$  provides an approximation of the posterior distribution  $p(\theta|y)$ . ABC rejection sampling thus provides one with a means of approximating the posterior distribution without having to ever evaluate the likelihood function explicitly. In many cases, the algorithm is run continuously until a total of  $k$  samples have been accepted. The value of  $k$  will vary for each dataset, but typically several choices are  $k$  will be used until the output is satisfactory.

The major problem with this formulation of the algorithm is that it tends to suffer from the curse of dimensionality for continuous data, and the size of  $\epsilon$  will subsequently have to increase in order

for the algorithm to be effective. Hence, Tavaré *et al.*, (1997) proposed the use of summary statistics so that distance evaluation in step 3 of the algorithm is replaced with

$$\|S(\mathbf{y}^*) - S(\mathbf{y})\| < \epsilon \quad (2.39)$$

where  $S(\cdot)$  denotes the summary statistic.

We consider now a simplified example of this algorithm to demonstrate its effectiveness. We will work with the situation where it is possible to determine both the likelihood function and the posterior distribution exactly to be able to compare the results of the algorithm and evaluate its performance.

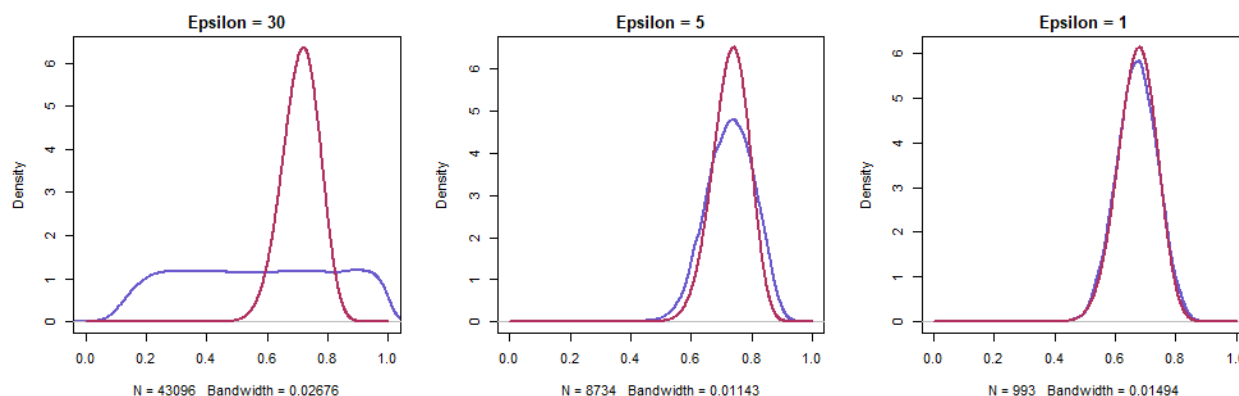
We consider the case of Bernoulli trials with a total of  $x$  successes over  $N = 100$  trials which leads to a binomial likelihood function with a beta conjugate prior. The posterior distribution is again a beta distribution, as shown below:

$$\begin{aligned} p(\theta|y) &\propto p(y|\theta)p(\theta) \\ &\propto \theta^x(1-\theta)^{N-x}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{x+\alpha-1}(1-\theta)^{N-x+\beta-1}. \end{aligned} \quad (2.40)$$

The parameters for the posterior distribution are hence given by  $\alpha' = x + \alpha$  and  $\beta' = N - x + \beta$ . We will suppose that the true probability of successes is given by  $\theta = 0.7$  and use a uniform prior distribution for this parameter i.e. a  $Beta(1,1)$  distribution. The aim of the ABC rejection sampling algorithm is to obtain an approximation of this posterior distribution without having to evaluate the likelihood function directly. A sufficient statistic for the binomial distribution is given by  $x$  which represents the total number of successes. In the case of binary data where a success is a 1, the sufficient statistic simply is given by

$$S(\mathbf{y}) = \sum_{i=1}^N y_i. \quad (2.41)$$

The algorithm was executed for a total of 50 000 iterations, using values of  $\epsilon = 30, 5, 1$ . These results are displayed in Figure 2.5. The red curve represents the true posterior distribution which, in this case, can be determined exactly since we assumed a conjugate prior was appropriate. The blue curve represents the estimated density function based on the accepted samples from the ABC algorithm.



**Figure 2.5: Results of the ABC rejection sampling algorithm.**

For large values of  $\epsilon$ , too many of the sampled parameter values are kept, and all we essentially obtain is a replication of the prior distribution. For example, when  $\epsilon = 30$ , we accepted just over 86% of the proposed samples. The approximated posterior distribution improves significantly as the value of  $\epsilon$  decreases, and it is almost identical to the true posterior distribution when  $\epsilon = 1$ . This corresponds to approximately 2% of the proposed sample values being accepted. Although a trivial example, it highlights the effectiveness of the ABC rejection sampling algorithm.

## 2.4 SUMMARY

The main elements of probability theory discussed in this chapter provide the background for most of the techniques used throughout this thesis. Probability theory is fundamental to many machine learning approaches, be it Bayesian or frequentist, and most of the derivations that will be shown are merely a repeated application of these simple and commonly encountered probability theory concepts.

Many Bayesian methods encountered involve the evaluation of complex integrals that may not exist in closed form or are computationally infeasible to evaluate in closed form. This historically left Bayesian methods side-lined. However, the advent of improved computing power prompted an increase in the use of Bayesian methods and allowed for easier use of Monte Carlo sampling techniques. This meant that efficient and accurate estimation of complex integrals could be performed without intensive computational power needed. As demonstrated, MCMC methods are a simple but powerful tool for Bayesian analysis and many of the examples used in this thesis will rely on them for approximations.



## CHAPTER 3

### BAYESIAN APPROACHES TO CLASSIFICATION

#### 3.1 DISCRIMINATE VERSUS GENERATIVE MODELS

The Bayesian approach to classification is one which is conceptually simple to understand since it is in line with one's intuition: classify a new observation to the most probable class. This means that when using the Bayesian approach to classification, we need to determine the posterior probabilities for each possible class. Assuming we have a dataset with input variable  $x$  and response  $Y$ , where  $Y$  belongs to any one of  $K$  classes, a new observation  $x$  is classified according to the rule

$$\text{Assign } x \text{ to } k^* = \underset{k}{\operatorname{argmax}} p(Y = k|x), \quad \text{for } k = 1, \dots, K. \quad (3.1)$$

Using Bayes' theorem, we can express these posterior probabilities as

$$p(Y = k|x) = \frac{p(x|Y = k)p(Y = k)}{p(x)}, \quad k = 1, \dots, K \quad (3.2)$$

and since  $p(x)$  is constant over each class assignment, we can equivalently express our Bayesian classification rule as

$$\text{Assign } x \text{ to } k^* = \underset{k}{\operatorname{argmax}} p(x|Y = k)p(Y = k), \quad k = 1, \dots, K \quad (3.3)$$

since it is not the numerical value of the maximum we are interested, but rather the class that leads to the maximum posterior probability.

The prior probabilities,  $p(Y = k)$ ,  $k = 1, \dots, K$  express our beliefs (or uncertainties) about the classes before we observe any data observations. Once sample data is obtained, we update these priors through multiplication by the likelihood which is exactly the expression obtained above in 3.2. If the distribution from which the data arose was known and these quantities could be evaluated exactly, this would result in the optimal Bayes classifier, and no other classifier would do better than this one. Obviously, in practice, we do not know the true underlying distribution of the observed data and thus other classification methods are used to approximate this optimal Bayes decision boundary as accurately as possible.

Classification models are either generative or discriminative depending on what the focus of the modelling approach is. In a discriminative learning model, the input data  $x$  is not directly modelled to a specific distribution; rather, the focus is on specifically modelling the dependency between the inputs and output variable. Thus, discriminative models aim to determine the posterior probabilities directly, without making any distributional assumptions. These models can be more simply regarded as ones that focus on constructing a decision boundary between classes.

In contrast, when the distribution of the input data is used, we obtain what is known as generative modelling. This then results in there being an associated probabilistic distribution for each of the possible classes, and these distributions are learned using different methods. For these models, the focus is on modelling the distributions from which the data arose and then using this to deduce to which classes new observations should be classified (Theodoridis, 2015:63).

This chapter will consider two methods of classification – logistic regression and naïve Bayes. These two approaches to classification arise commonly in the literature; however, the Bayesian approach to each of these tends to receive less attention. Hence, these two methods will be considered and compared with their Bayesian counterparts. To construct a naïve Bayes model, the focus is on modelling the joint distribution of the inputs and the response and then making use of Bayes rule to determine the posterior probabilities of the classes to make prediction; hence, it is a generative modelling approach. On the other hand, when performing logistic regression, since only the posterior probabilities of the data are of interest and not the actual distribution of the data, this is an example of a discriminative modelling approach.

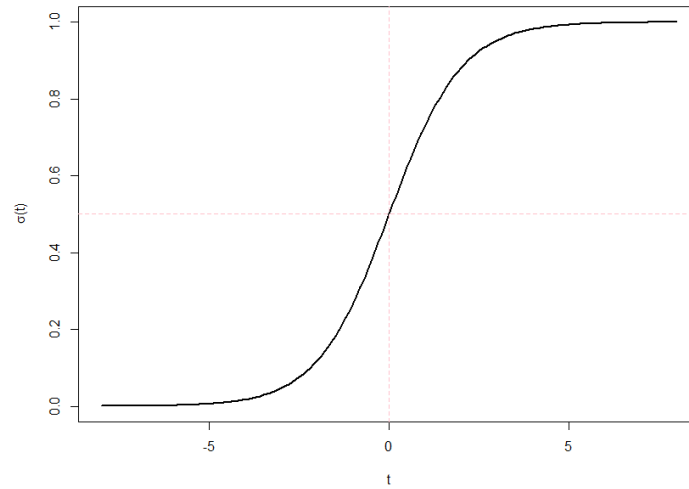
### 3.2 LOGISTIC REGRESSION

The aim behind the logistic regression model is to directly model the posterior probabilities using the respective conditional density functions. As mentioned, since the exact distribution of the data is not of interest, this is an example of a discriminate classifier (Theodoridis, 2015:290).

For simplicity, we consider the case of binary classification where the response variable  $Y$  takes on values according to  $Y \in \{0,1\}$ . As will be seen, the case of multiclass logistic regression extends analogously. Based on a set of observed inputs,  $\mathbf{x}$ , we model the posterior probabilities as

$$\begin{aligned}
 p(Y = 1|\mathbf{x}) &= \frac{p(\mathbf{x}|Y = 1)p(Y = 1)}{p(\mathbf{x}|Y = 1)p(Y = 1) + p(\mathbf{x}|Y = 0)p(Y = 0)} \\
 &= \frac{1}{1 + \frac{p(\mathbf{x}|Y = 0)p(Y = 0)}{p(\mathbf{x}|Y = 1)p(Y = 1)}} \\
 &= \frac{1}{1 + \exp\left(-\log\left(\frac{p(\mathbf{x}|Y = 1)p(Y = 1)}{p(\mathbf{x}|Y = 0)p(Y = 0)}\right)\right)} \\
 &= \frac{1}{1 + e^{-t}}, \quad \text{where } t = \log\left(\frac{p(\mathbf{x}|Y = 1)p(Y = 1)}{p(\mathbf{x}|Y = 0)p(Y = 0)}\right) \\
 &= \sigma(t).
 \end{aligned} \tag{3.4}$$

Here,  $\sigma(t)$  is an s-shaped function and is thus referred to as the sigmoid function (Bishop, 2006: 197). This sigmoid function is plotted in Figure 3.1.



**Figure 3.1: Logistic Sigmoid function.**

To construct a logistic regression model, the posterior probabilities of the classes are modelled by the sigmoid function applied to a linear combination of the inputs. The motivation behind this model is to construct a model that is linear in  $x$  but ensures that the posterior probabilities for the classes still sum to 1 (Hastie, Tibshirani & Friedman, 2009:119).

Throughout, a linear model will be denoted by

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_p x_p + \varepsilon = \mathbf{x}^T \boldsymbol{\theta} + \varepsilon \quad (3.5)$$

for a response variable  $y$  related linearly to a predictor variable  $\mathbf{x} = [1, x_1, x_2, \dots, x_p]^T$  through the unknown parameter coefficients  $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_p]^T$ . The term  $\varepsilon$  represents the irreducible error, or random noise, in the model.

Hence, the binary logistic regression model is given by

$$p(Y = 1|\mathbf{x}) = \sigma(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}} \quad (3.6)$$

and, due to normalization,

$$\begin{aligned} p(Y = 0|\mathbf{x}) &= 1 - p(Y = 1|\mathbf{x}) \\ &= \frac{e^{-\boldsymbol{\theta}^T \mathbf{x}}}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}} \end{aligned} \quad (3.7)$$

where  $\boldsymbol{\theta}$  denotes the parameter vector for the model. This means that if we consider the log ratio of the posteriors, we obtain a linear function as desired:

$$\ln \left( \frac{p(Y = 1|\mathbf{x})}{p(Y = 0|\mathbf{x})} \right) = \log \left( \frac{\frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}}{\frac{e^{-\boldsymbol{\theta}^T \mathbf{x}}}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}} \right)$$

$$\begin{aligned}
 &= \log \left( \frac{1}{1 + e^{-\theta^T x}} \times \frac{1 + e^{-\theta^T x}}{e^{-\theta^T x}} \right) \\
 &= \log \left( \frac{1}{e^{-\theta^T x}} \right) \\
 &= \theta^T x \\
 &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p.
 \end{aligned} \tag{3.8}$$

This log ratio of the posterior is termed the log-odds and defines a hyperplane representing the decision boundary between the two classes. Values that fall on this decision boundary are those for which this ratio is equal to zero. The parameters for logistic regression problems can be solved using maximum likelihood estimation and the Newton Raphson iterative procedure (Bishop, 2006:205). As the name suggest, maximum likelihood (ML) methods find values of the parameters that maximize the probability of the data, i.e. the most probable values of the parameters based on that specific sample.

For the case of multiclass logistic regression, the response variable  $Y$  is no longer binary, but can take on any one of  $K$  possible classes, that is,  $Y \in \{1, 2, \dots, K\}$ . The logistic regression model extends directly to this multiclass case so that the model probabilities are defined by

$$p(Y = k|x) = \frac{e^{-\theta_k^T x}}{1 + \sum_{j=1}^{K-1} e^{-\theta_j^T x}}, \quad k = 1, 2, \dots, K \tag{3.9}$$

$$\begin{aligned}
 p(Y = K|x) &= 1 - \sum_{k=1}^{K-1} p(Y = k|x) \\
 &= \frac{1}{1 + \sum_{j=1}^{K-1} e^{-\theta_j^T x}}.
 \end{aligned} \tag{3.10}$$

It is straightforward to see that this model is equivalent to the binary case which occurs when  $K = 2$  and the classes are commonly labelled such that the response variable is given by the set  $Y \in \{0, 1\}$ . In the multiclass logistic regression case, the full parameter set  $\theta$  consists of the set of all parameter vectors  $\theta_k$  for each of the  $k = 1, \dots, K$  classes. Maximum likelihood estimation is again used to determine each of the model parameters.

One of the drawbacks of using the maximum likelihood method is that it returns a single point estimate for each of the parameter coefficients. Simply considering a point estimate does not provide any indication of the uncertainty associated with the estimate and this is an important aspect to obtain to measure the reliability of the estimate. Although it is possible to obtain an estimate of the standard error associated with this estimate, a Bayesian approach to logistic regression provides an alternative. A Bayesian approach to logistic regression results in obtaining

a full posterior distribution for each of the parameters and this distribution will then immediately reflect our uncertainty in the estimates (Barber, 2011:389).

### 3.2.1 Bayesian Logistic Regression

This section now considers a Bayesian approach to logistic regression. Performing exact Bayesian inference for logistic regression would require the full determination of the posterior distribution over the parameters, that is, determining  $p(\boldsymbol{\theta}|\mathcal{D})$  where  $\mathcal{D}$  denotes the dependence on the data. However, this cannot be done since no convenient conjugate prior exists for the logistic regression model and so exact Bayesian inference is intractable. This will be seen in Equation 3.20 to follow where the log of the posterior distribution is derived. The Laplace method of approximating the posterior distribution using a Gaussian distribution is one of the methods that can be used to obtain an estimate of the posterior.

### 3.2.2 Laplace Approximation

The Laplacian approximation of a density function works by locally approximating any probability density function in terms of a Gaussian distribution centred at the most probable value. It is a convenient approximation method to use since it is typically quick to apply, fairly easy to understand and tends to perform accurately enough in general (Theodoridis, 2015:598).

We firstly define the energy function as  $En(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}, \mathcal{D})$  where  $p(\boldsymbol{\theta}, \mathcal{D})$  denotes the joint distribution function of the parameter vector  $\boldsymbol{\theta}$  and the observed data  $\mathcal{D}$ . Representing the normalization constant as  $Z = p(\mathcal{D})$ , we can express the posterior distribution as

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) &= \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} \\ &= \frac{1}{p(\mathcal{D})} e^{\log p(\boldsymbol{\theta}, \mathcal{D})} \\ &= \frac{1}{Z} e^{-En(\boldsymbol{\theta})}. \end{aligned} \tag{3.11}$$

We then perform a Taylor series expansion of the energy function around the most probable point, say,  $\boldsymbol{\theta}^*$ , since this corresponds to the lowest energy state (i.e. the minimum of the energy function will correspond to the maximum of the posterior),

$$En(\boldsymbol{\theta}) \approx En(\boldsymbol{\theta}^*) + (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T En'(\boldsymbol{\theta}^*) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T En''(\boldsymbol{\theta}^*) (\boldsymbol{\theta} - \boldsymbol{\theta}^*). \tag{3.12}$$

Since  $\boldsymbol{\theta}^*$  is defined to be the mode of the distribution, the gradient at that point is zero. Hence,  $En'(\boldsymbol{\theta}^*) = \mathbf{0}$  and so we can write

$$En(\boldsymbol{\theta}) \approx En(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T En''(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \quad (3.13)$$

and substituting this into the posterior in Equation 3.11, we obtain a new density  $p^*(\boldsymbol{\theta}|\mathcal{D})$  that approximates  $p(\boldsymbol{\theta}|\mathcal{D})$  given by

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) \approx p^*(\boldsymbol{\theta}|\mathcal{D}) &= \frac{1}{Z} e^{-[En(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T En''(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*)]} \\ &= \frac{1}{Z} e^{-En(\boldsymbol{\theta}^*)} \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)\right] \end{aligned} \quad (3.14)$$

where  $\mathbf{H} = En''(\boldsymbol{\theta}^*)$  is known as the Hessian matrix.

Thus, we see that

$$p^*(\boldsymbol{\theta}|\mathcal{D}) \sim \mathcal{N}(\boldsymbol{\theta}^*, \mathbf{H}^{-1}) \quad (3.15)$$

with

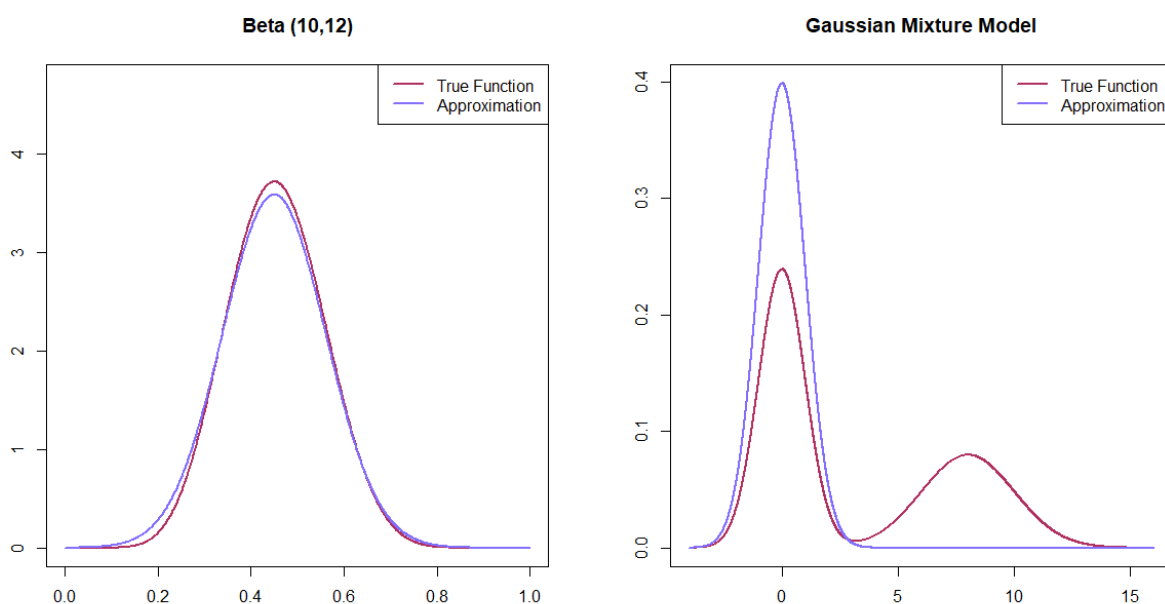
$$\begin{aligned} Z &= p(\mathcal{D}) \\ &\approx \int p^*(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \\ &= (2\pi)^{p/2} |\mathbf{H}|^{-\frac{1}{2}} e^{-En(\boldsymbol{\theta}^*)} \end{aligned} \quad (3.16)$$

which follows since  $Z$  is the normalizing constant of a multivariate Gaussian distribution. This distribution now represents a Gaussian approximation to the posterior distribution. As the sample size of the observed data increases, this approximation will usually improve since posterior distributions tend to take on a more Gaussian shape for larger  $N$  (Murphy, 2012:255). Naturally, those distributions which are similar in form to a Gaussian (i.e. a unimodal bell-curve) will have better Laplace approximations than those which look nothing like a Gaussian distribution.

This newly defined Gaussian distribution will be well-defined as long as the covariance matrix  $\mathbf{H}^{-1}$  (or the precision matrix  $\mathbf{H}$ ) is positive definite. Positive definiteness means that the point  $\boldsymbol{\theta}^*$  is indeed a local maximum and not instead a local minimum or saddle point (both of which will also have zero second derivatives). This local maximum is typically found using a numerical optimization technique. In the case of multimodal data, different Laplace distributions will be obtained depending on which mode is used. Clearly, this leads to a major drawback of the Laplace approximation method: some global properties of the distribution to be approximated might be missed since we consider only localized values in the method (Bishop, 2006: 215).

Figure 3.2 below shows a simple example of a Laplace approximation to a Beta density function as well as a Gaussian mixture model. The approximation is clearly more accurate for the Beta(10,12) distribution since the target distribution is very similar in shape to a Gaussian

distribution. In the case of the Gaussian mixture model, the approximation is poor since multimodality cannot be captured in a Laplace approximation.



**Figure 3.2: Laplace approximation of a beta density function (left) and a Gaussian mixture model (right).**

In cases where the Laplace approximation is poor, several other methods exist to better capture the global nature of the distribution. One common method implemented is variational logistic regression (Jaakkola & Jordan, 1997). This method also leads to a Gaussian approximation of the posterior distribution by maximizing a lower bound for the marginal likelihood function. Hence, the problem is transformed into an optimization approach – either minimization or maximization – by introducing extra variables into the model which are referred to as variational parameters.

Variational logistic regression can yield an approximation with higher accuracy due to increased flexibility in the approach but is more complex than the Laplace approximation method. Monte Carlo techniques are another popular alternative used to obtain an estimate of the posterior distribution and tend to be more commonly used in practice than Laplace approximation since they are applicable more generally.

This Laplace, or Gaussian, approximation can now be applied to the logistic regression scenario to obtain estimates of the posterior distributions for a Bayesian approach (Murphy, 2012:256). Since we seek a Gaussian approximation, we begin with a Gaussian prior with fixed hyperparameters  $\mu_0$  and  $V_0$  to respectively denote the prior mean and covariance matrix,

$$p(\theta) \sim \mathcal{N}(\mu_0, V_0). \quad (3.17)$$

The posterior distribution is then given by

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \quad (3.18)$$

where  $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$ . Let  $p(x_n) = P(Y = 1|x_n)$ . Since we are working with a binary response variable and the output of the model provides the probability of the class allocation being to class 1, we use the likelihood function of a Bernoulli distribution. The likelihood function is given by

$$\begin{aligned} p(\mathbf{y}|\boldsymbol{\theta}) &= \prod_{n=1}^N p(x_n)^{y_n} (1 - p(x_n))^{1-y_n} \\ \Rightarrow \log p(\mathbf{y}|\boldsymbol{\theta}) &= \sum_{n=1}^N \{y_n \log p(x_n) + (1 - y_n) \log(1 - p(x_n))\}. \end{aligned} \quad (3.19)$$

Using this likelihood function and the log of the Gaussian prior given in Equation 3.17, the log of the posterior distribution can be written as

$$\begin{aligned} \log p(\boldsymbol{\theta}|\mathbf{y}) &= \left[ \sum_{n=1}^N \{y_n \log p(x_n) + (1 - y_n) \log(1 - p(x_n))\} \right] \\ &\quad + \left[ \log \left( |\mathbf{V}_0|^{-\frac{1}{2}} (2\pi)^{-\frac{N}{2}} \right) - \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \mathbf{V}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_0) \right] \\ &= \left[ \sum_{n=1}^N \{y_n \log p(x_n) + (1 - y_n) \log(1 - p(x_n))\} \right] \\ &\quad + \left[ -\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \mathbf{V}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_0) + c \right] \end{aligned} \quad (3.20)$$

where  $c = \log \left( |\mathbf{V}_0|^{-\frac{1}{2}} (2\pi)^{-\frac{N}{2}} \right)$  denotes the normalizing constant.

The distribution given in Equation 3.20 is clearly not normally distributed, and inspection of the log likelihood function confirms that no conjugate prior is possible for the logistic regression and so a method of estimation is now required. To obtain a Gaussian approximation to this, we need to determine the maximum of this distribution, which we will denote by  $\boldsymbol{\theta}^{\max}$ . This can be done using any numerical optimizer. Since we are only interested in the unnormalized log posterior, the Hessian is then given by

$$\mathbf{H} = - \frac{\partial^2 \log p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{\max}} = - \frac{\partial^2 \log p(\boldsymbol{\theta}|\mathbf{y})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{\max}} \quad (3.21)$$

which is derived below. We have

$$\begin{aligned} \log p(x_n) &= \log P(Y = 1|x_n). \\ &= \log \left( \frac{1}{1 + e^{-\boldsymbol{\theta}^T x}} \right) \\ &= \log(1) - \log(1 + e^{-\boldsymbol{\theta}^T x}) \end{aligned}$$



$$= -\log(1 + e^{-\theta^T x}). \quad (3.22)$$

Hence,

$$\frac{\partial}{\partial \theta} \log p(\mathbf{x}_n) = -\frac{-e^{-\theta^T \mathbf{x}_n} \mathbf{x}_n}{1 + e^{-\theta^T \mathbf{x}_n}} = (1 - p(\mathbf{x}_n)) \mathbf{x}_n. \quad (3.23)$$

Also,

$$\begin{aligned} \log(1 - p(\mathbf{x}_n)) &= \log\left\{\frac{e^{-\theta^T \mathbf{x}_n}}{1 + e^{-\theta^T \mathbf{x}_n}}\right\} \\ &= -\theta^T \mathbf{x}_n - \log(1 + e^{-\theta^T \mathbf{x}_n}). \end{aligned} \quad (3.24)$$

Thus,

$$\begin{aligned} \frac{\partial}{\partial \theta} \log(1 - p(\mathbf{x}_n)) &= -\mathbf{x}_n - \frac{-e^{-\theta^T \mathbf{x}_n} \mathbf{x}_n}{1 + e^{-\theta^T \mathbf{x}_n}} \\ &= \mathbf{x}_n + (1 - p(\mathbf{x}_n)) \mathbf{x}_n \\ &= -\mathbf{x}_n p(\mathbf{x}_n). \end{aligned} \quad (3.25)$$

So, we have

$$\begin{aligned} &\frac{\partial}{\partial \theta} \left[ \sum_{n=1}^N \{y_n \log p(\mathbf{x}_n) + (1 - y_n) \log(1 - p(\mathbf{x}_n))\} \right] \\ &= \sum_{n=1}^N \{y_n (1 - p(\mathbf{x}_n)) \mathbf{x}_n + (1 - y_n) (-\mathbf{x}_n p(\mathbf{x}_n))\} \\ &= \sum_{n=1}^N \{y_n \mathbf{x}_n - y_n p(\mathbf{x}_n) \mathbf{x}_n - \mathbf{x}_n p(\mathbf{x}_n) + y_n \mathbf{x}_n p(\mathbf{x}_n)\} \\ &= \sum_{n=1}^N \{y_n \mathbf{x}_n - \mathbf{x}_n p(\mathbf{x}_n)\} \\ &= \sum_{n=1}^N \mathbf{x}_n (y_n - p(\mathbf{x}_n)). \end{aligned} \quad (3.26)$$

Thus, for the second derivative, we get

$$\frac{\partial}{\partial \theta^T} \sum_{n=1}^N \{y_n \mathbf{x}_n - \mathbf{x}_n p(\mathbf{x}_n)\} = -\sum_{n=1}^N \frac{\partial}{\partial \theta^T} (\mathbf{x}_n p(\mathbf{x}_n)). \quad (3.27)$$

Now,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}^T} (x_n p(x_n)) &= \frac{\partial}{\partial \boldsymbol{\theta}^T} \left[ \frac{x_n}{1 + e^{-\boldsymbol{\theta}^T x_n}} \right] \\
&= \frac{(1 + e^{-\boldsymbol{\theta}^T x_n})(\mathbf{0}) - x_n(-e^{-\boldsymbol{\theta}^T x_n} x_n^T)}{[1 + e^{-\boldsymbol{\theta}^T x_n}]^2} \\
&= \frac{x_n(e^{-\boldsymbol{\theta}^T x_n} x_n^T)}{[1 + e^{-\boldsymbol{\theta}^T x_n}]^2} \\
&= \frac{x_n(e^{-\boldsymbol{\theta}^T x_n} x_n^T)}{[1 + e^{-\boldsymbol{\theta}^T x_n}]^2} \\
&= \frac{(e^{-\boldsymbol{\theta}^T x_n}) x_n x_n^T}{[1 + e^{-\boldsymbol{\theta}^T x_n}]^2} \cdot \frac{1}{1 + e^{-\boldsymbol{\theta}^T x_n}} \\
&= x_n x_n^T (1 - p(x_n)) p(x_n).
\end{aligned} \tag{3.28}$$

Also, it is easy to see that

$$\frac{\partial^T}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \left( -\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \mathbf{V}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_0) + c \right) = -\mathbf{V}_0^{-1}. \tag{3.29}$$

Thus, the Hessian is given by

$$\begin{aligned}
\mathbf{H} &= - \frac{\partial^2 \log p(\boldsymbol{\theta} | \mathbf{y})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \Big|_{\boldsymbol{\theta} = \boldsymbol{\theta}^{max}} \\
&= \mathbf{V}_0^{-1} + \sum_{n=1}^N p(x_n)(1 - p(x_n)) x_n x_n^T
\end{aligned} \tag{3.30}$$

using  $\boldsymbol{\theta} = \boldsymbol{\theta}^{max}$  to calculate the probabilities  $p(x_n), n = 1, \dots, N$ .

Thus, the Gaussian approximation to this posterior distribution, using the Laplace approximation method, is given by

$$\hat{p}(\boldsymbol{\theta} | \mathbf{y}) \sim \mathcal{N}(\boldsymbol{\theta}^{max}, \mathbf{V}_N) \tag{3.31}$$

where the covariance matrix is given by  $\mathbf{V}_N = \mathbf{H}^{-1}$ . Here,  $\boldsymbol{\theta}^{max}$  is in fact equal to the maximum *a posteriori* (MAP) estimator which will be discussed in the next chapter (Bishop, 2006:218).

This posterior distribution can then be used to derive the predictive distribution, for which Monte Carlo approximation is typically used to approximate the integrals, and predictions can be made based on new inputs. Given a new input,  $\mathbf{x}^*$ , the posterior predictive distribution, as introduced in Section 2.1.1, is given by

$$p(y^* = 1 | \mathbf{x}^*, \mathcal{D}) = \int p(y^* = 1 | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta}$$

$$\approx \int p(\mathbf{x}^*) \mathcal{N}(\boldsymbol{\theta}^{max}, \mathbf{V}_N) d\boldsymbol{\theta} \quad (3.32)$$

since we made a Gaussian approximation to the posterior. Analytical determination of this integral is typically not possible, and so further approximation is required. This integral is generally approximated using Monte Carlo sampling, so that the posterior predictive distributed is estimated by

$$p(y^* = 1 | \mathbf{x}^*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^S \sigma(\boldsymbol{\theta}_s^T \mathbf{x}^*) \quad (3.33)$$

where each  $\boldsymbol{\theta}_s$  is generated from  $\mathcal{N}(\boldsymbol{\theta}^{max}, \mathbf{V}_N)$ ,  $s = 1, \dots, S$  (Murphy, 2012: 258).

### 3.2.3 Applications

This section will provide two examples of Bayesian Logistic regression. The first example is derived from the one provided in Murphy (2012: 254) based on a linearly separable simulated dataset and illustrates the Laplace approximation to the posterior. The second example uses the well-known South African heart disease dataset, obtained from the ElemStatLearn R package (Hastie *et al.*, 2019), and compares the performance of classical logistic regression to Bayesian logistic regression as well as using Laplace approximation to the posterior versus a Monte Carlo approximation.

#### 3.2.3.1 Simulated Data

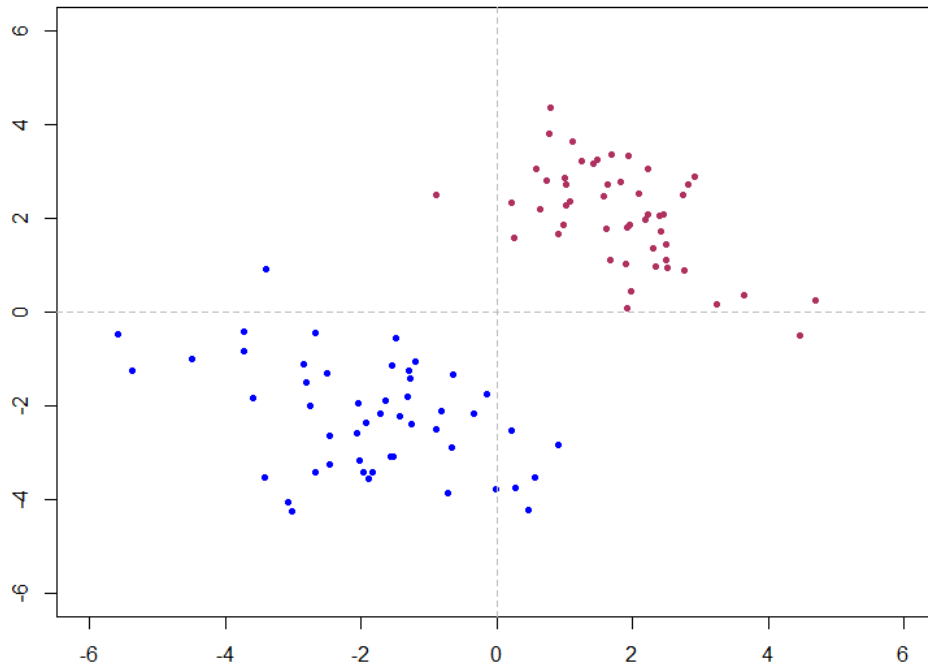
The data used for this example was simulated as follows: the mean vectors for class 1 and class 2 are given respectively by

$$\boldsymbol{\mu}_1 = [2, 2]^T \quad \text{and} \quad \boldsymbol{\mu}_2 = [-2, -2]^T \quad (3.34)$$

and the associated covariance matrices are given by

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1.5 & -1 \\ -1 & 1.5 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}. \quad (3.35)$$

The data are shown in Figure 3.3 with class 1 represented by the maroon points and class 2 points represented by blue points.



**Figure 3.3: Simulated dataset.**

The prior distribution for the parameters was set as a zero-mean Gaussian distribution with a variance of 100 for each parameter. That is,

$$p(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{V}_0) \quad (3.36)$$

where

$$\boldsymbol{\mu}_0 = [0, 0]^T \quad \text{and} \quad \mathbf{V}_0 = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}. \quad (3.37)$$

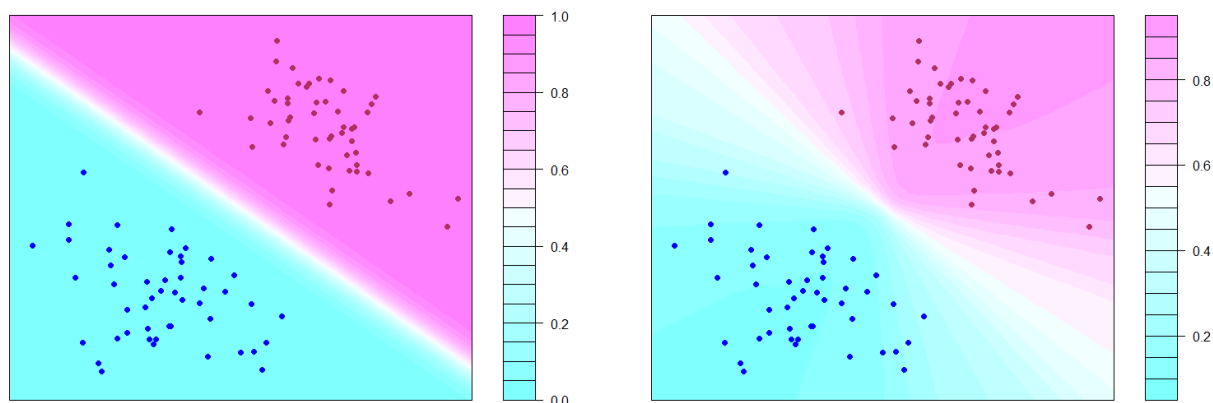
This prior is essentially non-informative, with only a small influence being placed on the parameters taking on values close to zero (i.e. being small). Using this prior and the logistic regression likelihood function, the posterior distribution was estimated through the use of the Laplace approximation. The resulting Gaussian approximation to the posterior has a mean vector given by

$$\boldsymbol{\theta}^{\max} = [2.7030, 3.2563]^T \quad (3.38)$$

and covariance matrix

$$\mathbf{V}_N = \begin{bmatrix} 13.1231 & 0.17978 \\ 0.17978 & 12.9645 \end{bmatrix}. \quad (3.39)$$

The results are shown in Figure 3.4 below.



**Figure 3.4: The decision boundary using a point estimate (left) and the decision boundary using MC averaging (right).**

The left-hand plot in Figure 3.4 shows the filled contours for the decision boundary when using only a point estimate. In this case, the decision boundary was constructed using only the distribution of  $\theta^{\max}$  to determine the probability of a point belonging to class 1, i.e. the maroon points. The right-hand plot shows the filled contours obtained using an MC average over 50 samples from the posterior predictive distribution. Even though each sample from the posterior predictive distribution results in a linear decision boundary, averaging over them results in the posterior predictive density function no longer being linear. Instead, the decision boundary spreads out more and more the further one moves away from the training data, showing the increased uncertainty in the decision boundary (Murphy, 2012:259).

### 3.2.3.2 South African Heart Disease Data

A common dataset used as an introduction for fitting logistic regression models is the South African Heart Disease dataset. This dataset contains  $N = 462$  observations and  $p = 7$  predictors all relating to different indicators that put an individual at risk of heart disease. The final column of the dataset is a binary variable 'chd' denoting whether that individual showed a presence of heart disease,  $\text{chd} = 1$ , or not,  $\text{chd} = 0$ .

The dataset was split into a training set (70%) and a held-out test set (30%) and two main analyses were done:

1. A classical logistic regression model was fit to the training set and an estimate of the test error was obtained by applying this model to the held-out test dataset.
2. A Bayesian logistic regression model was fit to the training dataset and an estimate of the test error was obtained by applying this model to the test dataset. The posterior distribution was estimated by:
  - a. Monte Carlo simulations.

## b. Laplace approximation.

Table 3.1 shows the comparative point estimates for the parameter coefficients obtained from the classical and Bayesian approaches.

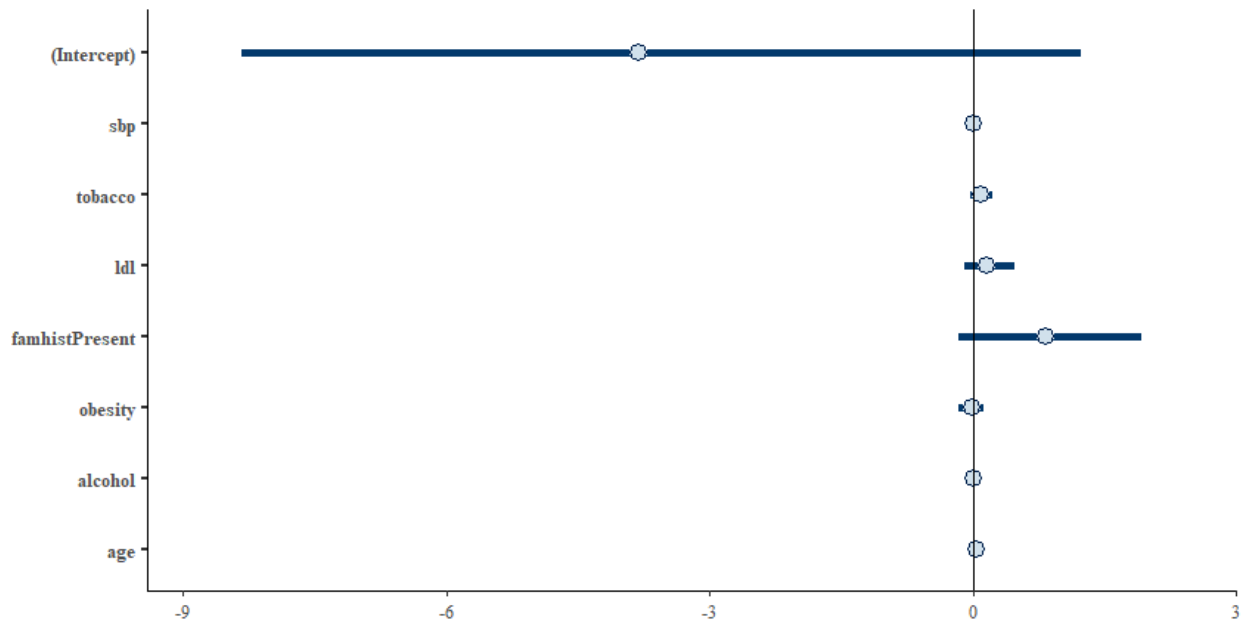
**Table 3.1: Parameter coefficient estimates.**

Variable name	Classical Logistic regression	Bayesian Logistic Regression: MCMC	Bayesian Logistic Regression: Laplace Approximation
Intercept	−3.751353	−3.794023	−0.633798
sbp	0.002748	0.002826	−0.000410
tobacco	0.083548	0.087303	0.098407
ldl	0.153695	0.158741	0.239395
famhistPresent	0.808761	0.825870	0.567010
obesity	−0.015263	−0.017091	−0.111715
alcohol	0.001140	0.001004	−0.002938
age	0.037171	0.037824	0.025173

The point estimates obtained from the classical logistic regression approach are determined via maximum likelihood estimation and the Newton-Raphson algorithm, and the point-estimates for the Bayesian approach using MCMC are given by the median of the posterior sample of the Monte Carlo simulations. The median is commonly used as a point estimate from MCMC simulations since this coincides with the construction of credible intervals in Bayesian analyses and medians also tend to be less sensitive to outliers than means.

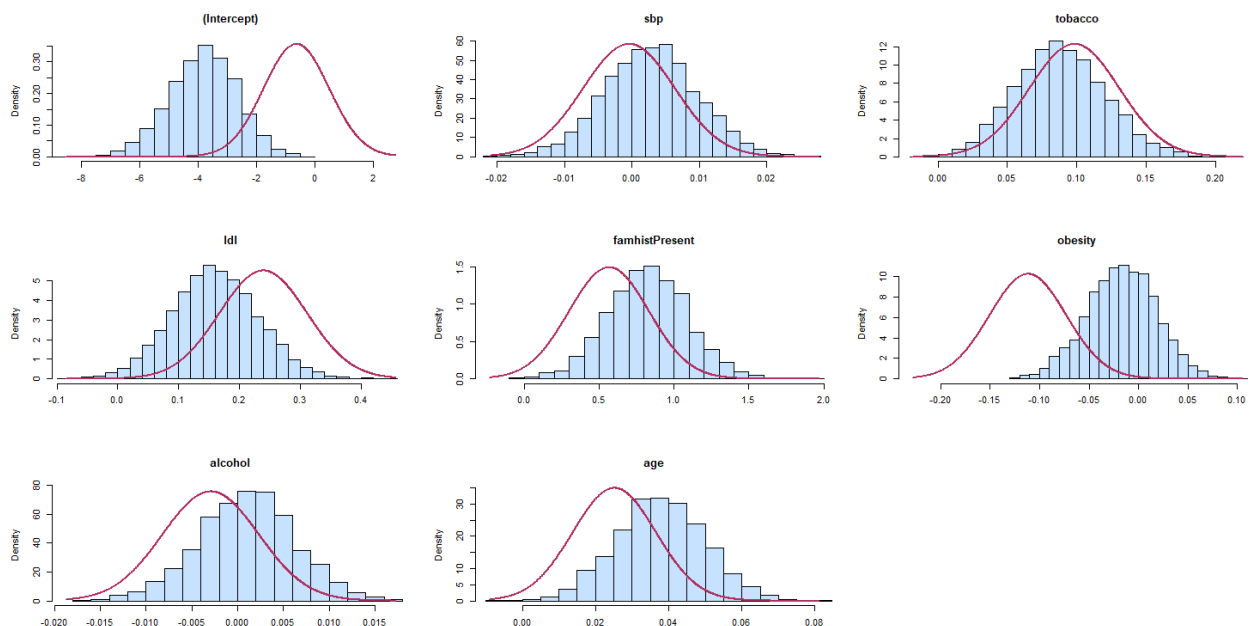
The point estimates obtained from the classical logistic regression approach and the Bayesian approach using MCMC do not differ by much, with slightly more variability occurring for the point estimates obtained from the Laplace approach, but it is due to the fact that we are able to obtain estimates of the full posterior distribution over the parameters in the Bayesian approach that make the method more appealing. This dataset was also not very large which makes Bayesian implementations as easy and efficient to implement as the frequentist approach. Figure 3.5 displays the estimated posterior distribution over each of the parameters obtained using Monte Carlo simulations. These posterior distributions are all shown together in the form of an interval with the length of the interval reflecting the posterior variability of each parameter. It is immediate to see from this plot which variables have a greater associated uncertainty. Since variable selection techniques will only be discussed in Chapter 5, this step of performing variable selection

and reducing the total number of parameters in the model was not performed and so all variables were included in the model.



**Figure 3.5: Posterior distributions of the parameters using MCMC.**

To further visualize the form of each posterior distribution, Figure 3.6 shows the individual histograms of the resulting MCMC samples. All the posterior distributions appear to be approximately normally distributed; that is, they are individually roughly symmetric with a bell-shape curve. Overlaid with the histograms of MCMC samples are the marginal densities obtained using a Laplace approximation.



**Figure 3.6: Histograms of the MCMC samples for each parameter overlaid with the marginal density function for each parameter obtained using Laplace approximation.**

Table 3.2 below summarizes the test errors obtained from the different approaches.

**Table 3.2: Test errors obtained using the three approaches to logistic regression.**

	<b>Classical Logistic regression</b>	<b>Bayesian Logistic Regression: MCMC</b>	<b>Bayesian Logistic Regression: Laplace Approximation</b>
<b>Test error</b>	0.2733813	0.2589928	0.2661871

Both Bayesian approaches outperformed the classical logistic regression approach in this setting which shows the benefits of modelling the parameters as random variables to directly incorporate the uncertainty associated with them. It is not surprising that the Laplace approximation performed well overall since it appears from the histograms obtained from the MCMC sampling that all of the posterior distributions of the parameters do seem to be approximately Gaussian and Laplace approximation performs best in these situations. The Laplace approximation was also faster to implement than the MCMC although had a slightly larger test error. However, faster implementation with only minimal effect on the test error is often a reason to favour Laplace approximations over MCMC, especially when the dataset is large, or many MCMC iterations are required. Even though the Bayesian approaches only slightly improved the test error of the classical approach, the prior distribution placed over the parameters was non-informative. If there was more specific knowledge available about the parameters prior to the analysis, a more informative prior could have been used which should naturally lead to an even greater improvement in the Bayesian approaches.

### 3.3 NAÏVE BAYES

Naïve Bayes is a popular approach that has remained in use for many years despite being a suboptimal classifier (Hastie *et al.*, 2009: 210). We consider the situation of making a classification based on a set of inputs  $x$ . If we assume that the predictors are conditionally independent given the class labels for the response variable, we then obtain a naïve Bayes classifier. The suboptimality stems from the fact that this independence assumption results in a significantly simplified model, but it is exactly this simplicity that makes the naïve Bayes approach popular.

Suppose that the observed input variable is  $p$ -dimensional,  $x = [x_1, x_2, \dots, x_p]^T$ , and each of the attributes,  $x_i$ , takes on  $s_i$  possible values, for  $i = 1, \dots, p$ . The naïve Bayes assumption states that the inputs are conditionally independent given the class, that is,

$$p(x|Y = k) = \prod_{i=1}^p p(x_i|Y = k), \quad \text{for all } k = 1, \dots, K \quad (3.40)$$



where  $k = 1, \dots, K$  are the possible class assignments.

Applying Bayes' rule, we can write the posterior distribution as

$$\begin{aligned} p(Y = k|\mathbf{x}) &= \frac{p(\mathbf{x}|Y = k)p(Y = k)}{p(\mathbf{x})} \\ &= \frac{p(Y = k) \prod_{i=1}^p p(x_i|Y = k)}{p(\mathbf{x})}. \end{aligned} \quad (3.41)$$

Since the above denominator will be constant for each class, we need only consider

$$f_k(\mathbf{x}) := p(Y = k|\mathbf{x}) \propto p(Y = k) \prod_{i=1}^p p(x_i|Y = k) \quad (3.42)$$

and the resulting naïve Bayes classifier is given by (Bishop, 2006: 380)

$$f^{NB}(\mathbf{x}) = \underset{k}{\operatorname{argmax}} f_k(\mathbf{x}), \quad k = 1, \dots, K. \quad (3.43)$$

The term 'naïve' comes from the fact that, in real life, this independence assumption will almost always be wrong since nothing in nature is truly independent of each other. However, the motivation behind the use of the method is that it is a much simpler and easier problem to handle when the assumption of independence is made (Murphy, 2012:82).

As an example of how the naïve Bayes modelling approach can be applied, we will consider the Sentiment Labelled Sentences dataset from the UCI Machine Learning Repository (Kotzias *et al.*, 2015). This dataset consists of a total of 3000 labelled reviews of products, movies and restaurants coming from three different website sources: imdb.com, amazon.com and yelp.com. Each of these websites contain 500 positive reviews and 500 negative reviews. A naïve Bayes classifier can be used to determine whether a particular review has a positive (1) or negative (0) sentiment, by assuming that all words in the review are conditionally independent of each other given the class label. This, of course, demonstrates the naivety of the approach since there surely must exist some form of dependence between words that occur when someone is writing a positive or negative review. Nevertheless, it will be shown that the classifier still performs fairly well despite this oversimplification.

### 3.3.1 Maximum Likelihood Estimation

We recall that the each of the posterior distribution functions within the naïve Bayes classifier can be expressed as

$$p(Y = k|\mathbf{x}) \propto p(Y = k) \prod_{i=1}^p p(x_i|Y = k) \quad (3.44)$$

where  $k = 1, \dots, K$  represents one of the class labels associated with the input vector. To evaluate the naïve Bayes classifier, we need to obtain estimates for each of the probabilities defined in the model. It is these probabilities which constitute the model parameters. This means that we need to obtain estimates for  $p(Y = k)$  for all  $k$ , as well as estimates for the conditional probabilities  $p(x_i|Y = k)$  for each of the entries  $i$  in the observed vector  $x$  for all classes. Note, that due to normalization, not all these probabilities will need to be estimated. For example, since we know that the sum over all prior class probabilities must be equal to 1, that is,

$$\sum_{k=1}^K p(Y = k) = 1 \quad (3.45)$$

we only need to obtain the estimates for  $p(Y = k), k = 1, \dots, K - 1$  since the remaining  $p(Y = K)$  can be obtained from the rest using by solving for  $P(Y = k)$ :

$$\begin{aligned} \sum_{k=1}^K p(Y = k) &= 1 \\ \Rightarrow p(Y = K) &= 1 - \sum_{k=1}^{K-1} p(Y = k). \end{aligned} \quad (3.46)$$

Maximum likelihood estimation can be used to obtain estimates for these above-mentioned parameters. For simplicity, we consider the case of multivariate Bernoulli naïve Bayes, where  $Y \in \{1, 2, \dots, K\}$  and we suppose that our predictor variable  $\mathbf{X} = [X_1, X_2, \dots, X_p]^T$  is a  $p$ -dimensional Bernoulli random variable, that is, each  $X_i \in \{0, 1\}$ .

Consider the independent and identically distributed (i.i.d) training set  $\{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$ , where  $\mathbf{x}_n = [x_{n1}, x_{n2}, \dots, x_{np}]^T$  with  $x_{ni} \in \{0, 1\}$  for  $i = 1, \dots, p$  and  $y_n$  denotes the class label corresponding to  $\mathbf{x}_n$ , with  $y_n \in \{1, 2, \dots, K\}$ . Since each attribute in the observed vector follows a Bernoulli distribution, it follows that

$$X_i|Y = k \sim \text{Bernoulli}(\mu_{ik}) \quad (3.47)$$

where  $\mu_{ik} = p(X_i = 1|Y = k)$ . Thus, for any one observation  $\mathbf{x}$ , it follows that

$$\begin{aligned} p(\mathbf{X} = \mathbf{x}|Y = k) &= \prod_{i=1}^p p(x_i = 1|Y = k)^{x_i} (1 - p(x_i = 1|Y = k))^{1-x_i} \\ &= \prod_{i=1}^p \mu_{ik}^{x_i} (1 - \mu_{ik})^{1-x_i} \end{aligned} \quad (3.48)$$

where  $\mu_{ik} = p(x_i = 1|Y = k)$ .

When the data are i.i.d, the likelihood function can be expressed as the product

$$L = \prod_{n=1}^N p(x_n, Y_n = k) \quad (3.49)$$

and so, the log-likelihood, denoted by  $\ell$ , is given by

$$\begin{aligned} \ell &= \sum_{n=1}^N \log p(x_n, Y_n = k) \\ &= \sum_{n=1}^N \log \left\{ p(Y_n = k) \prod_{i=1}^p p(x_{ni} | Y_n = k) \right\} \\ &= \sum_{n=1}^N \log \left\{ \pi_k \prod_{i=1}^p \mu_{ik}^{x_{ni}} (1 - \mu_{ik})^{1-x_{ni}} \right\} \\ &= \sum_{n=1}^N \left\{ \log \pi_k + \log \left( \prod_{i=1}^p \mu_{ik}^{x_{ni}} (1 - \mu_{ik})^{1-x_{ni}} \right) \right\} \\ &= \sum_{n=1}^N \left\{ \log \pi_k + \sum_{i=1}^p \log \left\{ \mu_{ik}^{x_{ni}} (1 - \mu_{ik})^{1-x_{ni}} \right\} \right\} \\ &= \sum_{n=1}^N \left\{ \log \pi_k + \sum_{i=1}^p \{x_{ni} \log \mu_{ik} + (1 - x_{ni}) \log(1 - \mu_{ik})\} \right\} \end{aligned} \quad (3.50)$$

where  $\pi_k = P(Y_n = k)$ , such that  $\sum_{k=1}^K \pi_k = 1$ .

To determine the maximum likelihood estimates, we differentiate  $\ell$  with respect to  $\mu_{ik}$ , set equal to zero and solve. Thus,

$$\begin{aligned} \frac{\partial \ell}{\partial \mu_{ik}} &= \sum_{n=1}^N I(Y_n = k) \left\{ \frac{x_{ni}}{\mu_{ik}} - \frac{1 - x_{ni}}{1 - \mu_{ik}} \right\} \\ &= \sum_{n=1}^N I(Y_n = k) \{x_{ni}(1 - \mu_{ik}) - \mu_{ik}(1 - x_{ni})\} \\ &= \sum_{n=1}^N I(Y_n = k) \{x_{ni} - x_{ni}\mu_{ik} - \mu_{ik} + \mu_{ik}x_{ni}\}. \end{aligned} \quad (3.51)$$

Hence,

$$\begin{aligned} \frac{\partial \ell}{\partial \mu_{ik}} &= 0 \\ \Rightarrow \sum_{n=1}^N I(Y_n = k) \{x_{ni} - x_{ni}\hat{\mu}_{ik} - \hat{\mu}_{ik} + \hat{\mu}_{ik}x_{ni}\} &= 0 \end{aligned}$$

$$\begin{aligned} \Rightarrow \sum_{n=1}^N I(Y_n = k)(x_{ni}) &= \hat{\mu}_{ik} \sum_{n=1}^N I(Y_n = k) \{x_{ni} + 1 - x_{ni}\} \\ \Rightarrow \hat{\mu}_{ik} &= \frac{\sum_{n=1}^N I(Y_n = k)(x_{ni})}{\sum_{n=1}^N I(Y_n = k)}. \end{aligned} \quad (3.52)$$

This makes intuitive sense, since it corresponds to the number of times  $x_i = 1$  for class  $k$ , divided by the total number of datapoints in class  $k$ .

We use Lagrange multipliers to find the MLE for  $\pi_k = P(Y_n = k)$  using the constraint  $\sum_{k=1}^K \pi_k = 1$ .

$$\begin{aligned} \frac{\partial \ell}{\partial \pi_k} + \lambda \frac{\partial \sum_k \pi_k}{\partial \pi_k} &= 0 \\ \Rightarrow \sum_{n=1}^N \frac{I(Y_n = k)}{\pi_k} + \lambda &= 0 \\ \Rightarrow \lambda &= - \sum_{n=1}^N \frac{I(Y_n = k)}{\pi_k} \end{aligned}$$

and so

$$\pi_k = - \sum_{n=1}^N \frac{I(Y_n = k)}{\lambda}. \quad (3.53)$$

Further, using the constraint  $\sum_k \pi_k = 1$ , we have

$$\begin{aligned} 1 &= \sum_{k=1}^K \pi_k = - \sum_{k=1}^K \sum_{n=1}^N \frac{I(Y_n = k)}{\lambda} \\ \Rightarrow \lambda &= - \sum_{k=1}^K \sum_{n=1}^N I(Y_n = k) = -N. \end{aligned} \quad (3.54)$$

Hence,

$$\hat{\pi}_k = \sum_{n=1}^N \frac{I(Y_n = k)}{N} \quad (3.55)$$

which is again what we would intuitively expect: our estimate for the proportion in class  $k$  is simply given by the total number of observations in class  $k$  divided by the sample size (Barber, 2011:242).

### 3.3.2 Naïve Bayes Variants

We recall that the naïve Bayes classifier is given by

$$f^{NB}(\mathbf{x}) = \underset{k}{\operatorname{argmax}} p(Y = k) \prod_{i=1}^p p(x_i|Y = k), \quad k = 1, \dots, K. \quad (3.56)$$

In the previous section, the case of Bernoulli naïve Bayes was considered whereby the predictor variables took on only binary attributes. Other variants of the naïve Bayes classifier exist, such as the Gaussian naïve Bayes and the multinomial naïve Bayes. These variants arise from the desire to model the conditional distribution of the predictor variables in a different manner.

When the predictor variables take on real-valued entries, the Gaussian naïve Bayes classifier can be used whereby the class conditional densities are now modelled by a normal distribution, that is, we assume for each predictor  $X_i$ ,

$$X_i|Y = k \sim \mathcal{N}(\mu_{ik}, \sigma_{ik}^2) \quad (3.57)$$

where  $\mu_{ik}$  denotes the mean value of the  $j$ th attribute in the predictor  $X_i$  for class  $k$  and  $\sigma_{ik}^2$  is the associated variance. Thus,

$$\begin{aligned} p(\mathbf{x}|Y = k) &= \prod_{i=1}^p p(x_i|Y = k) \\ &= \prod_{i=1}^p \frac{1}{\sigma_{ik}\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma_{ik}^2}(x_i - \mu_{ik})^2\right). \end{aligned} \quad (3.58)$$

The parameter estimates can be obtained in an analogous manner as before.

Another common scenario is when the attributes take on values in the set  $x_i \in \{1, 2, \dots, S\}$  and the response variable  $Y$  falls into 1 of  $K$  possible classes, that is,  $Y \in \{1, 2, \dots, K\}$ . We then model the conditional densities using a multinomial distribution:

$$p(\mathbf{x}|Y = k) = \binom{N}{x_1 \dots x_p} \prod_{i=1}^p \mu_{ik}^{x_i} \quad (3.59)$$

where  $\sum_{i=1}^p x_i = N$ . This model is commonly used for document classification, where each attribute in the predictor variable represents the number of times that a particular word occurs in a document, and  $N$  represents the document length.

The maximum likelihood estimator for the multinomial distribution was derived in Section 2.1.3 and again in Section 3.3.1 in the naïve Bayes context. However, there are now more parameters in this model since the attributes are no longer binary and so we require estimates for the probability of the  $i$ th attribute in class  $k$  being in state  $s$ :

$$\mu_{ik,s} = p(x_i = s|Y = k), \quad i \in \{1, \dots, p\}, s \in \{1, \dots, S\} \text{ and } k \in \{1, \dots, K\}. \quad (3.60)$$

It follows analogously from Equation 3.52 that

$$\hat{\mu}_{ik,s} = \frac{\sum_{n=1}^N I(x_{ni} = s)I(Y_n = k)}{\sum_{n=1}^N \sum_{s=1}^S I(x_{ni} = s)I(Y_n = k)} \quad (3.61)$$

which is simply the relative number of times that attribute  $i$  takes on a value  $s$  for class  $k$ . The MLE for the class probabilities is the same as before, given by

$$\hat{\pi}_k = \sum_{n=1}^N \frac{I(Y_n = k)}{N} \quad (3.62)$$

for each class  $k = 1, \dots, K$ .

### 3.3.3 Application: Sentiment Analysis

A naïve Bayes classifier will now be applied to the Sentiment Labelled Sentences dataset to predict whether a review has a positive or negative sentiment. All reviews were pre-processed before the model was fit: all letters were reduced to lower case (so that the word ‘Good’ and ‘good’ are equivalent in the analysis); all numbers and punctuation were removed; all common stop-words were removed (such as “the” or “a”); and lastly, any excess whitespaces, such as from a tab or double space, were stripped down to a single whitespace. Table 3.3 shows an example of five reviews before and after the processing.

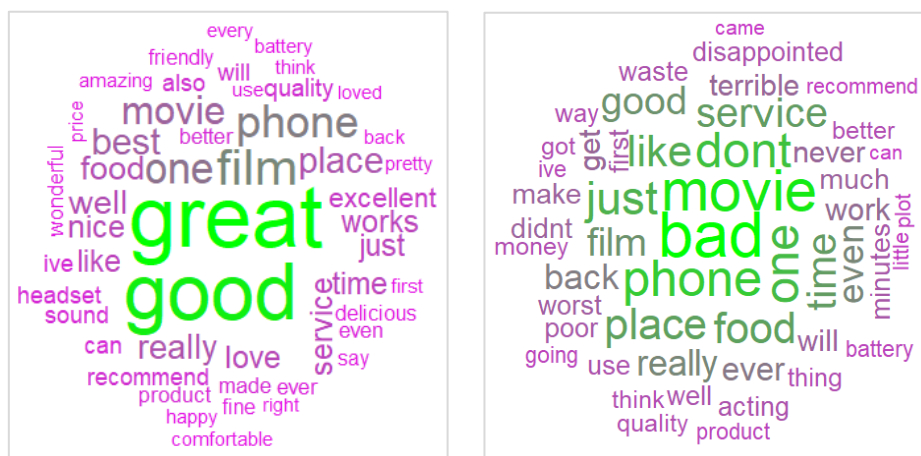
**Table 3.3: Comparison of raw test review and cleaned text review.**

Raw Review	Review after cleaning
“So there is no way for me to plug it in here in the US unless I go by a converter.”	“way plug us unless go converter”
“Good case, Excellent value.”	“good case excellent value”
“Great for the jawbone.”	“great jawbone”
“Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!”	“tied charger conversations lasting minutesmajor problems”
“The mic is great.”	“mic great”

Furthermore, to reduce any chances of overfitting and to reduce the total number of parameters in the model, only those terms which appeared in at least 5 different reviews were considered. A matrix was then constructed where each row referred to a review, and each column indicated either a 1 if that term appeared in the review, or a zero if it did not. For example, for the first review given in Table 3.3, the matrix would contain a row of zeros, with ones occurring only in the columns corresponding to the terms ‘way’, ‘plug’, ‘us’, ‘unless’, ‘go’ and ‘converter’.

A 70/30 training and test sample was randomly selected and the corresponding rows in the full matrix of reviews was split to create a subsequent training and a test set. This means that the

training and test set had the same number of columns corresponding to the same set of words. The word clouds in Figure 3.7 below show the 50 most frequently occurring words for positive and negative sentiments in the training set, respectively.



**Figure 3.7: Word clouds for positive sentiments (left) and the negative sentiments (right).**

A Bernoulli naïve Bayes classifier was fit to the training data and the resulting confusion matrix is given in Table 3.4 showing the performance of the model when applied to the held-out test dataset.

**Table 3.4: Confusion matrix for the Bernoulli naïve Bayes classifier.**

		True values	
		Negative	Positive
Predicted values	Negative	369	83
	Positive	89	359

Hence, the overall test error for this fit is given by

$$\frac{83 + 89}{83 + 89 + 369 + 359} = \frac{172}{900} = 19.11\%. \quad (3.63)$$

### 3.3.4 Bayesian Naïve Bayes

There are a few notable drawbacks with the naïve Bayes modelling approach, the most notable being the very strong assumption of conditional independence. Despite this strong assumption, the naïve Bayes approach still tends to perform quite well in applications hence its continued popularity. Another drawback of the approach is the use of maximum likelihood as a means of obtaining parameter estimates. Maximum likelihood estimation is well-known to be prone to overfitting and a possible solution to this is the use of a fully Bayesian approach to naïve Bayes.

This approach assigns prior distributions to each of the parameters in the model to discourage any extreme values in the confidence of predictions (i.e. 0 or 1). These extreme values occur quite often in classical naïve Bayes approaches. Another possible solution to this problem of extreme predictions is to smooth the resulting probabilities. This can be achieved by adding a small frequency count to each of the attributes so that none of them are exactly zero – just very small. A special case of smoothing actually results as a specific instance of the Bayesian approach as will be shown.

To explain further and practically demonstrate this issue of overconfident predictions, we return to the sentiment analysis done previously. Even though we removed the words which did not appear in more than 5 documents, there is still an issue that arises. Table 3.5 gives the probabilities of assigning a new data point to each class if it contains the attribute word ‘unless’ as well as the probabilities of each class if it contains the attribute word ‘blue’.

**Table 3.5: Class conditional probabilities.**

	<b>Negative</b>	<b>Positive</b>
‘unless’	0.004798464	0.000000000
‘blue’	0.000000000	0.002835539

The reason for the extreme probabilities of zero can be understood by considering only those rows in the training data matrix that correspond to positive reviews and then looking at the column for the word ‘unless’. This column will contain only zeros since none of the documents in the training data in that class set contained the word ‘unless’. Hence, the model makes the extreme conclusion that any review containing the word ‘unless’ will never be a positive review. Similarly, any review containing the word ‘blue’ will never be classified into the negative sentiment class since none of the training samples in the negative class contain the word ‘blue.’ This problem occurs for 83 words in the positive class and 64 words in the negative class. This clearly highlights a major drawback of the classical naïve Bayes modelling approach.

Using a Bayesian approach, we can predict the output class,  $y$ , of an input  $x$  as follows,

$$p(y|x, \mathcal{D}) \propto p(x|\mathcal{D}, y)p(y|\mathcal{D}) \quad (3.64)$$

where  $\mathcal{D}$  denotes the observed data. Hence, we require the specification of the class conditional density function to obtain the likelihood function of the data as well as the specification of a prior distribution for the parameters in the model. We will continue working with the Bernoulli naïve Bayes variant where the predictors are binary to align with the sentiment analysis example.



Often, to simplify matters, the class probabilities are set through maximum likelihood estimation, so we use the same result as previously, that is, we set

$$\hat{\pi}_k = \sum_{n=1}^N \frac{I(Y_n = k)}{N} = \frac{N_k}{N} \quad (3.65)$$

for each class  $k$  where  $N_k = \sum_{n=1}^N I(Y_n = k)$ , and so we do not assume any distribution for these parameters.

Alternatively, we could also assign a prior distribution to these class probabilities to ensure a fully Bayesian analysis. Since  $Y$  is a discrete, unordered, random variable, we can model its distribution as a categorical distribution, so that the likelihood function is given by

$$p(\mathbf{y}|\boldsymbol{\pi}) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{I(Y_n=k)} = \prod_{k=1}^K \pi_k^{N_k}. \quad (3.66)$$

Further, since the Dirichlet distribution is conjugate to the categorical distribution, we can assign a Dirichlet prior to the probabilities  $\pi_k = p(Y = k)$ , that is,

$$p(\boldsymbol{\pi}) \sim \text{Dirichlet}(\boldsymbol{\pi}; \boldsymbol{\beta}) \propto \prod_{k=1}^K \pi_k^{\beta_k - 1} \quad (3.67)$$

and so, it follows that the posterior distribution for the class probabilities is given by

$$\begin{aligned} p(\boldsymbol{\pi}|\mathbf{y}) &\propto p(\mathbf{y}|\boldsymbol{\pi})p(\boldsymbol{\pi}) \\ &= \prod_{k=1}^K \pi_k^{N_k} \prod_{k=1}^K \pi_k^{\beta_k - 1} \\ &= \prod_{k=1}^K \pi_k^{N_k + \beta_k - 1} \\ &= \prod_{k=1}^K \pi_k^{\beta'_k - 1} \end{aligned} \quad (3.68)$$

which is again a Dirichlet distribution, with the  $k$ th entry of the parameter vector given by  $\beta'_k = \beta_k + N_k$ . Here,  $\beta_k$  is treated as a hyperparameter and values for this hyperparameter will be considered shortly.

Further, we require a prior for each of the  $\mu_{ik}, i = 1, \dots, p$  and  $k = 1, \dots, K$  which represent the class conditional probabilities, that is,  $\mu_{ik} = p(X_i = 1|Y = k)$ . We will set each of these priors to be a  $\text{Beta}(\alpha_0, \alpha_1)$  distribution where the specific values for  $\alpha_0$  and  $\alpha_1$  will be considered later. Thus, the prior for  $\boldsymbol{\mu} = \{\mu_{ik}, i = 1, \dots, p, k = 1, \dots, K\}$  is given by

$$p(\boldsymbol{\mu}) = \prod_{i=1}^p \prod_{k=1}^K p(\mu_{ik}) \quad (3.69)$$

where

$$p(\mu_{ik}) \propto \mu_{ik}^{\alpha_0-1} (1 - \mu_{ik})^{\alpha_1-1}. \quad (3.70)$$

Thus, using the Bernoulli class conditional likelihood function, the posterior distribution is given by

$$\begin{aligned} p(\mu_{ik}|\mathcal{D}) &\propto p(\mathcal{D}|\mu_{ik})p(\mu_{ik}) \\ &\propto \left\{ \prod_{n=1}^N \mu_{ik}^{I(Y_n=k)(x_{ni})} (1 - \mu_{ik})^{I(Y_n=k)(1-x_{ni})} \right\} \mu_{ik}^{\alpha_0-1} (1 - \mu_{ik})^{\alpha_1-1} \\ &= \mu_{ik}^{\sum_{n=1}^N I(Y_n=k)(x_{ni})} (1 - \mu_{ik})^{\sum_{n=1}^N I(Y_n=k)(1-x_{ni})} \mu_{ik}^{\alpha_0-1} (1 - \mu_{ik})^{\alpha_1-1} \\ &= \mu_{ik}^{N_{ik}} (1 - \mu_{ik})^{N_k - N_{ik}} \mu_{ik}^{\alpha_0-1} (1 - \mu_{ik})^{\alpha_1-1} \\ &= \mu_{ik}^{N_{ik} + \alpha_0 - 1} (1 - \mu_{ik})^{\alpha_1 + N_k - N_{ik} - 1} \end{aligned} \quad (3.71)$$

which is again a Beta distribution. Hence,

$$p(\mu_{ik}|\mathcal{D}) \sim \text{Beta}(\alpha_0 + N_{ik}, \alpha_1 + (N_k - N_{ik})) \equiv \text{Beta}(\alpha'_0, \alpha'_1) \quad (3.72)$$

where

$$N_{ik} = \sum_{n=1}^N I(Y_n = k) (x_{ni}). \quad (3.73)$$

Returning to the naïve Bayes modelling approach, we recall that the aim is to determine

$$\begin{aligned} f^{NB}(\mathbf{x}) &= \underset{k}{\operatorname{argmax}} p(Y = k|\mathbf{x}), \quad k = 1, \dots, K \\ &= \underset{k}{\operatorname{argmax}} p(Y = k) \prod_{i=1}^p p(x_i|Y = k), \quad k = 1, \dots, K. \end{aligned} \quad (3.74)$$

For a fully Bayesian approach, we need to compute the posterior predictive distribution and integrate out the unknown parameter values. Given a new observation  $\mathbf{x}^*$ , we assign it to a class  $k^*$  where

$$k^* = \underset{k}{\operatorname{argmax}} p(Y = k) \prod_{i=1}^p p(x_i^*|Y = k). \quad (3.75)$$

As mentioned,  $p(Y = k)$  is often simply set using maximum likelihood estimation. Another common approach is to set this value to be equal for each class, that is  $p(Y = k) = \frac{1}{K}$  for all classes. We consider here the fully Bayesian approach where the class probabilities are also

assigned a prior distribution and this prior distribution is that which was given in Equation 3.67. Using a fully Bayesian approach to naïve Bayes, we assign a new observation  $\mathbf{x}^*$  to the class  $k^*$ , where,

$$\begin{aligned}
 k^* &= \operatorname{argmax}_k p(Y = k | \mathbf{x}^*, \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
 &= \operatorname{argmax}_k \iint p(Y = k, \boldsymbol{\pi}, \boldsymbol{\mu} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}) d\boldsymbol{\pi} d\boldsymbol{\mu} \\
 &= \operatorname{argmax}_k \iint p(Y = k | \boldsymbol{\pi}) p(\boldsymbol{\pi} | \mathcal{D}, \boldsymbol{\beta}) p(\mathbf{x}^* | \boldsymbol{\mu}) p(\boldsymbol{\mu} | \mathcal{D}, \boldsymbol{\alpha}) d\boldsymbol{\pi} d\boldsymbol{\mu} \\
 &= \operatorname{argmax}_k \left\{ \int p(Y = k | \boldsymbol{\pi}) p(\boldsymbol{\pi} | \mathcal{D}, \boldsymbol{\beta}) d\boldsymbol{\pi} \right\} \left\{ \int \prod_{i=1}^p p(x_i^* | \boldsymbol{\mu}) p(\boldsymbol{\mu} | \mathcal{D}, \boldsymbol{\alpha}) d\boldsymbol{\mu} \right\} \\
 &= \operatorname{argmax}_k \left\{ \int p(Y = k | \boldsymbol{\pi}) p(\boldsymbol{\pi} | \mathcal{D}, \boldsymbol{\beta}) d\boldsymbol{\pi} \right\} \operatorname{argmax}_k \left\{ \int \prod_{i=1}^p p(x_i^* | \boldsymbol{\mu}) p(\boldsymbol{\mu} | \mathcal{D}, \boldsymbol{\alpha}) d\boldsymbol{\mu} \right\}. \quad (3.76)
 \end{aligned}$$

Firstly,

$$\begin{aligned}
 \operatorname{argmax}_k \int p(Y = k | \boldsymbol{\pi}) p(\boldsymbol{\pi} | \mathcal{D}, \boldsymbol{\beta}) d\boldsymbol{\pi} &= \operatorname{argmax}_k \int \operatorname{Cat}(y, \boldsymbol{\pi}) \operatorname{Dirichlet}(\boldsymbol{\pi}; \boldsymbol{\beta}') d\boldsymbol{\pi} \\
 &= \operatorname{argmax}_k \int \prod_{k=1}^K \pi_k^{I(Y=k)} \left( \frac{1}{B(\boldsymbol{\beta}')} \right) \prod_{k=1}^K \pi_k^{\beta'_k - 1} d\boldsymbol{\pi} \\
 &= \operatorname{argmax}_k \int \left( \frac{1}{B(\boldsymbol{\beta}')} \right) \prod_{k=1}^K \pi_k^{\beta'_k + I(Y=k) - 1} d\boldsymbol{\pi} \\
 &= \operatorname{argmax}_k \left( \frac{1}{B(\boldsymbol{\beta}')} \right) \int \prod_{k=1}^K \pi_k^{\beta'_k + I(Y=k) - 1} d\boldsymbol{\pi} \\
 &= \operatorname{argmax}_k \frac{\Gamma(\sum_{k=1}^K \beta'_k)}{\prod_{k=1}^K \Gamma(\beta'_k)} \cdot \frac{\prod_{k=1}^K \Gamma(\beta'_k + I(Y = k))}{\Gamma(\sum_{k=1}^K (\beta'_k + I(Y = k)))} \\
 &= \operatorname{argmax}_k \frac{\Gamma(\sum_{k=1}^K (\beta_k + N_k))}{\prod_{k=1}^K \Gamma(\beta_k + N_k)} \cdot \frac{\prod_{k=1}^K \Gamma(\beta_k + N_k + I(Y = k))}{\Gamma(\sum_{k=1}^K (\beta_k + N_k + I(Y = k)))} \\
 &= \operatorname{argmax}_k \frac{\Gamma(\sum_{k=1}^K \beta_k + N)}{\prod_{k=1}^K \Gamma(\beta_k + N_k)} \cdot \frac{\prod_{k=1}^K \Gamma(\beta_k + N_k + I(Y = k))}{\Gamma(\sum_{k=1}^K \beta_k + N + 1)} \\
 &= \operatorname{argmax}_k \frac{\beta_k + N_k}{\sum_{k=1}^K \beta_k + N}. \quad (3.77)
 \end{aligned}$$

We now consider the second bracket in Equation 3.76. A useful result is that the posterior predictive distribution for a single observation is simply given by the posterior mean parameters, as shown below:

$$p(x_i = 1 | \mathcal{D}) = \int p(x_i = 1 | \mu_{ik}) p(\mu_{ik} | \mathcal{D}, \boldsymbol{\alpha}) d\mu_{ik}$$

$$\begin{aligned}
&= \int \mu_{ik} \text{Beta}(\alpha'_0, \alpha'_1) d\mu_{ik} \\
&= E[\mu_{ik}] \\
&= \frac{\alpha'_0}{\alpha'_0 + \alpha'_1}.
\end{aligned} \tag{3.78}$$

Thus,

$$\begin{aligned}
p(x_i = 0|\mathcal{D}) &= 1 - p(x_i = 1|\mathcal{D}) \\
&= 1 - \frac{\alpha'_0}{\alpha'_0 + \alpha'_1} \\
&= \frac{\alpha'_0 + \alpha'_1 - \alpha'_0}{\alpha'_0 + \alpha'_1} \\
&= \frac{\alpha'_1}{\alpha'_0 + \alpha'_1}.
\end{aligned} \tag{3.79}$$

Hence,

$$\begin{aligned}
\operatorname{argmax}_k \int \prod_{i=1}^p p(x_i^*|\mu_{ik}) p(\mu_{ik}|\mathcal{D}, \alpha) d\mu_{ik} &= \operatorname{argmax}_k \int \prod_{i=1}^p p(x_i^*|\mu_{ik}) p(\mu_{ik}|\mathcal{D}, \alpha) d\mu_{ik} \\
&= \operatorname{argmax}_k \int \prod_{i=1}^p \text{Bernoulli}(x_i^*; \mu_{ik}) \text{Beta}(\alpha'_0, \alpha'_1) d\mu_{ik} \\
&= \operatorname{argmax}_k \prod_{i=1}^p \left( \frac{\alpha'_0}{\alpha'_0 + \alpha'_1} \right)^{x_i^*} \left( \frac{\alpha'_1}{\alpha'_0 + \alpha'_1} \right)^{1-x_i^*} \\
&= \operatorname{argmax}_k \prod_{i=1}^p \left( \frac{N_{ik} + \alpha_0}{N_k + \alpha_0 + \alpha_1} \right)^{x_i^*} \left( \frac{N_k - N_{ik} + \alpha_1}{N_k + \alpha_0 + \alpha_1} \right)^{1-x_i^*} \\
&= \operatorname{argmax}_k \prod_{i=1}^p \frac{(N_{ik} + \alpha_0)^{x_i^*} (N_k - N_{ik} + \alpha_1)^{1-x_i^*}}{N_k + \alpha_0 + \alpha_1}.
\end{aligned} \tag{3.80}$$

Thus, a fully Bayesian approach to the Bernoulli naïve Bayes classifier would classify a new observation  $\mathbf{x}^*$  to the class  $k^*$  where,

$$k^* = \operatorname{argmax}_k \frac{\beta_k + N_k}{\sum_{k=1}^K \beta_k + N} \prod_{i=1}^p \frac{(N_{ik} + \alpha_0)^{x_i^*} (N_k - N_{ik} + \alpha_1)^{1-x_i^*}}{N_k + \alpha_0 + \alpha_1}. \tag{3.81}$$

Often,  $\beta$  and  $\alpha$  are all set equal to unity which corresponds to what is called add-one or Laplace smoothing and this choice of unity results in both priors being essentially non-informative (Murphy, 2012: 82; Xu, Li & Wang, 2017). In this case, the classification of a new point becomes

$$k^* = \underset{k}{\operatorname{argmax}} \frac{N_k + 1}{N + K} \prod_{i=1}^p \frac{(N_{ik} + 1)^{x_i^*} (N_k - N_{ik} + 1)^{1-x_i^*}}{N_k + 2}. \quad (3.82)$$

Returning to the sentiment analysis example, we now fit a Bayesian naïve Bayes classifier to the training data (in the form of add-one/Laplace smoothing). Table 3.6 shows the confusion matrix for the Bayesian classifier when applied to the same held-out test set.

**Table 3.6: Confusion matrix for the Bayesian naïve Bayes model.**

		True values	
		Negative	Positive
Predicted values	Negative	371	78
	Positive	87	364

The new test error using a Bayesian approach is

$$\frac{78 + 87}{371 + 78 + 87 + 364} = \frac{165}{900} = 18.33\% \quad (3.83)$$

which is a slight improvement over the 19.11% test error obtained from the classical naïve Bayes classifier.

Considering again the probabilities for reviews containing the attribute word ‘unless’ or ‘blue’, Table 3.7 summarizes the new probabilities for these attributes using the Bayesian approach.

**Table 3.7: Class conditional probabilities for Bayesian naïve Bayes.**

	Negative	Positive
‘unless’	0.005747126	0.0009433962
‘blue’	0.000957854	0.0037735849

There are no longer any zero probabilities for any of the attributes. The Bayesian approach ensures that smoothing occurs over the parameter estimates so that none of them are completely zero, just very small.

### 3.5 SUMMARY

Logistic regression and naïve Bayes approaches occur in many different fields of study where classification is the main aim of the analysis. Although not as popular and their classical counterparts, the Bayesian approaches to each of these methods provides a valuable advantage

in terms of full posterior distributions over the parameter estimates and a lower test error in each of the examples provided.

The Bayesian approach to logistic regression using MCMC resulted in a lower test error than the classical logistic regression even though the prior used was essentially non-informative. The Bayesian method using Laplace approximation was quick to implement and the Bayesian method using MCMC sampling was only slightly less fast since the dataset was not very big and so the MCMC simulations were quick enough to execute. Overall, the Bayesian approach to logistic regression gave superior results than classical logistic regression.

The Bayesian naïve Bayes modelling approach also improved the overall test error by reducing the extreme conclusions that can occur using a classical naïve Bayes approach. Despite being more complex mathematically, the Bayesian naïve Bayes approach was as efficient to implement practically. Thus, both examples provided in this chapter showed favour towards the Bayesian approaches in comparison with their frequentist versions based on the datasets used in each analysis.

## CHAPTER 4

### A BAYESIAN APPROACH TO LINEAR REGRESSION

#### 4.1 CLASSICAL MULTIPLE LINEAR REGRESSION

In a regression setting with multiple variables to consider, the aim is to model the relationship between a set of inputs, or predictors,  $x_1, x_2, \dots, x_p$  and an output, or response variable,  $y$ . Expressing this relationship as a linear regression model, we have

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_p x_p + \varepsilon = \mathbf{x}^T \boldsymbol{\theta} + \varepsilon \quad (4.1)$$

with  $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_p]^T$  and  $\mathbf{x} = [1, x_1, x_2, \dots, x_p]^T$ , with the first entry included in  $\mathbf{x}$  to account for the intercept term.

Hence, given a set of training data  $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$ , we can express the multiple linear regression model in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \quad (4.2)$$

where  $\mathbf{y} = [y_1, \dots, y_N]^T$ ,  $\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Np} \end{bmatrix}$ , and  $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_N]^T$  is a vector of random

noise, assumed to be i.i.d Gaussian random variables with a mean value of zero and a variance of  $\sigma_\varepsilon^2$ , that is,

$$\varepsilon_n \sim \mathcal{N}(0, \sigma_\varepsilon^2), \quad n = 1, \dots, N. \quad (4.3)$$

Since the random noise vector is unobservable, the aim in linear regression is to predict the value of the output variable given a new set of input variables as well as possible. This means finding the estimate  $\hat{\boldsymbol{\theta}}$  for  $\boldsymbol{\theta}$  that minimizes the squared error loss function. As a result, the well-known ordinary least-squares estimate is obtained, given by

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (4.4)$$

assuming the matrix  $\mathbf{X}^T \mathbf{X}$  is invertible – in which case the solution is unique.

#### 4.2 BAYESIAN LINEAR REGRESSION

In the classical linear regression scenario above, we determined a single set of fixed estimates for the unknown parameters. The Bayesian approach to linear regression differs by instead assuming that the unknown set of parameters are random variables that follow some distribution. This, however, does not assume that the parameters themselves display random behaviour, but rather allows for uncertainty in our estimate to be incorporated directly. This uncertainty is

included through the use of the prior distribution which represents our prior knowledge, or belief, about the possible values of the parameters before we have received any observations (Theodoridis, 2015: 586). Each time we receive a new observation, we update our prior belief and thus obtain a posterior distribution with a higher precision whenever more data is included into the model.

As mentioned in Section 2.2.2.1, for convenience, we rather work with the precision parameter  $\tau$  instead of the variance, that is, we assume  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \tau^{-1}\mathbf{I})$ . Further, we assume (for now) that  $\tau$  is a known constant, so that the model,  $y = \mathbf{x}^T \boldsymbol{\theta} + \varepsilon$ , has distribution

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(\mathbf{x}^T \boldsymbol{\theta}, \tau^{-1}) \equiv \sqrt{\frac{\tau}{2\pi}} \exp\left(-\frac{\tau}{2}(\mathbf{y} - \mathbf{x}^T \boldsymbol{\theta})^2\right). \quad (4.5)$$

Assuming that each observation is independent, the likelihood function of the data can be obtained by

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau) &= \prod_{n=1}^N p(y_n|\mathbf{x}_n, \boldsymbol{\theta}; \tau) \\ &= \prod_{n=1}^N \sqrt{\frac{\tau}{2\pi}} \exp\left(-\frac{\tau}{2}(y_n - \mathbf{x}_n^T \boldsymbol{\theta})^2\right) \\ &= \tau^{\frac{N}{2}} (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\tau}{2} \sum_{n=1}^N (y_n - \mathbf{x}_n^T \boldsymbol{\theta})^2\right) \\ &= \tau^{\frac{N}{2}} (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\tau}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right). \end{aligned} \quad (4.6)$$

We can then make use of Bayes' rule to obtain the posterior distribution for the model parameters. Hence, given some prior distribution,  $p(\boldsymbol{\theta})$ , the posterior distribution for the model parameters, denoted by  $p(\boldsymbol{\theta}|\mathbf{y})$ , is given by

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau)p(\boldsymbol{\theta})}{p(\mathbf{y})} \propto p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau)p(\boldsymbol{\theta}). \quad (4.7)$$

To obtain a posterior distribution that belongs to the same family of distributions as the prior distribution, we use a conjugate prior. Since the likelihood function follows a Gaussian distribution, using a Gaussian prior will result in a Gaussian posterior. The decision to use a conjugate prior in this Bayesian analysis, as well as others in practice, is purely for the sake of illustration and mathematical ease since we are then able to obtain a closed form solution for the posterior distribution. However, a poor choice of prior can lead to poor conclusions. A further discussion on the choice of prior distributions will be provided in Section 4.2.1.

Hence, we define the conjugate prior to be the following multivariate Gaussian distribution,



$$p(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \equiv \frac{1}{|\boldsymbol{\Sigma}_0|^{\frac{1}{2}}(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_0)\right) \quad (4.8)$$

where  $\boldsymbol{\mu}_0$  and  $\boldsymbol{\Sigma}_0$  denote the mean and covariance matrix, respectively, which for the moment are also assumed to be known. The posterior distribution is then derived using the result from Bayes rule in Equation 4.7 as follows:

$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{y}) &\propto p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau)p(\boldsymbol{\theta}) \\ &= \tau^{N/2}(2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\tau}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right) \frac{1}{|\boldsymbol{\Sigma}_0|^{\frac{1}{2}}(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_0)\right) \\ &= \frac{1}{Z} \exp\left\{-\frac{\tau}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_0)\right\} \end{aligned} \quad (4.9)$$

with  $\frac{1}{Z}$  representing the normalizing constant for the resulting posterior distribution. This is done since we are not immediately concerned with the full posterior, but rather just the unnormalized version.

The following result will be used to derive the posterior distribution for this example, as well as being applicable generally to any distributions of this form. Normal expansion of a Gaussian exponential term for the variable  $\mathbf{y}$  has the form

$$\begin{aligned} -\frac{\tau}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) &= -\frac{\tau}{2}(\mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X}\boldsymbol{\theta}) \\ &\propto -\frac{\tau}{2}(\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y}) \end{aligned} \quad (4.10)$$

with the proportionality relating to the consideration of only those terms containing the parameter vector  $\boldsymbol{\theta}$ . Hence, any exponential term that can be written in this form describes another Gaussian distribution with  $\boldsymbol{\theta}$  now as the variable of interest. Specifically, this is the same quadratic function of  $\boldsymbol{\theta}$  as for a Gaussian log density with covariance  $\tau^{-1}(\mathbf{X}^T \mathbf{X})^{-1}$  and mean  $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ . To verify this, we see that

$$\begin{aligned} &-\frac{1}{2}\{\boldsymbol{\theta} - (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\}^T \{\tau^{-1}(\mathbf{X}^T \mathbf{X})^{-1}\}^{-1} \{\boldsymbol{\theta} - (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\} \\ &= -\frac{\tau}{2}\{\boldsymbol{\theta} - (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\}^T \{\mathbf{X}^T \mathbf{X}\} \{\boldsymbol{\theta} - (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\} \\ &= -\frac{\tau}{2}\{\boldsymbol{\theta} - (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\}^T \{\mathbf{X}^T \mathbf{X}\boldsymbol{\theta} - \mathbf{X}^T \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\} \\ &= -\frac{\tau}{2}\{\boldsymbol{\theta} - (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\}^T \{\mathbf{X}^T \mathbf{X}\boldsymbol{\theta} - \mathbf{X}^T \mathbf{y}\} \\ &= -\frac{\tau}{2}\{\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X}\boldsymbol{\theta} + \mathbf{y}^T \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\} \\ &\propto -\frac{\tau}{2}\{\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X}\boldsymbol{\theta}\} \end{aligned}$$

$$= \frac{\tau}{2} \{ \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} \} \quad (4.11)$$

as desired.

Returning to the posterior distribution and using this above result, the exponential term in Equation 4.9 can be written as

$$\begin{aligned} & -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) - \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_0) \\ & = -\frac{\tau}{2} \{ \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \} - \frac{1}{2} \{ \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} - \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} + \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \}. \end{aligned}$$

Combining only the terms that involve  $\boldsymbol{\theta}$ , we have

$$\begin{aligned} & \frac{\tau}{2} \mathbf{y}^T \mathbf{X}\boldsymbol{\theta} + \frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} - \frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} + \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 + \frac{1}{2} \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} \\ & = \tau \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} - \frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} + \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 + \frac{1}{2} \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} \\ & = \tau \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} - \frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} + \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \\ & = -\frac{\tau}{2} \left\{ -2 \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \frac{1}{\tau} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\theta} - \frac{2}{\tau} \boldsymbol{\theta}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right\} \\ & = -\frac{\tau}{2} \left\{ \boldsymbol{\theta}^T \left[ \mathbf{X}^T \mathbf{X} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \right] \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \left[ \mathbf{X}^T \mathbf{y} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right] \right\}. \end{aligned} \quad (4.12)$$

Hence, this defines a Gaussian distribution for the variable  $\boldsymbol{\theta}$ , with precision matrix given by

$$\boldsymbol{\Sigma}_N^{-1} = \tau \left[ \mathbf{X}^T \mathbf{X} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \right] = \tau \mathbf{X}^T \mathbf{X} + \boldsymbol{\Sigma}_0^{-1} \quad (4.13)$$

and mean vector

$$\begin{aligned} \boldsymbol{\mu}_N & = \left\{ \mathbf{X}^T \mathbf{X} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \right\}^{-1} \left\{ \mathbf{X}^T \mathbf{y} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right\} \\ & = \left\{ \frac{1}{\tau} \boldsymbol{\Sigma}_N^{-1} \right\}^{-1} \left\{ \mathbf{X}^T \mathbf{y} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right\} \\ & = \boldsymbol{\Sigma}_N (\tau \mathbf{X}^T \mathbf{y} + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0). \end{aligned} \quad (4.14)$$

Completing the square in the exponent of Equation 4.12, gives

$$\begin{aligned} & -\frac{\tau}{2} \left\{ \boldsymbol{\theta}^T \left[ \mathbf{X}^T \mathbf{X} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \right] \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \left[ \mathbf{X}^T \mathbf{y} + \frac{1}{\tau} \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right] \right\} \\ & = -\frac{1}{2} (\boldsymbol{\theta}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N) \\ & = -\frac{1}{2} (\boldsymbol{\theta}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N + \boldsymbol{\mu}_N^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N) + \frac{1}{2} \boldsymbol{\mu}_N^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N \\ & = -\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_N)^T \boldsymbol{\Sigma}_N^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_N) + \frac{1}{2} \boldsymbol{\mu}_N^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N \end{aligned}$$

$$\propto -\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)^T \boldsymbol{\Sigma}_N^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_N). \quad (4.15)$$

Thus, the posterior has the form

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto \exp\left\{-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)^T \boldsymbol{\Sigma}_N^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)\right\} \quad (4.16)$$

which is exactly the form of an unnormalized Gaussian distribution (Bishop, 2006:152). Hence, if the likelihood function of the data is given by

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \tau^{-1}\mathbf{I}) \quad (4.17)$$

and the conjugate prior

$$p(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad (4.18)$$

is used, the resulting posterior distribution will be given by

$$p(\boldsymbol{\theta}|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N) \quad (4.19)$$

with

$$\boldsymbol{\Sigma}_N^{-1} = \tau \mathbf{X}^T \mathbf{X} + \boldsymbol{\Sigma}_0^{-1} \quad \text{and} \quad \boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N(\tau \mathbf{X}^T \mathbf{y} + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0). \quad (4.20)$$

This general result is applicable for all datasets that are modelled using a normal distribution with a conjugate prior and will be referred to several times throughout this thesis.

#### 4.2.1 The Prior vs The Likelihood

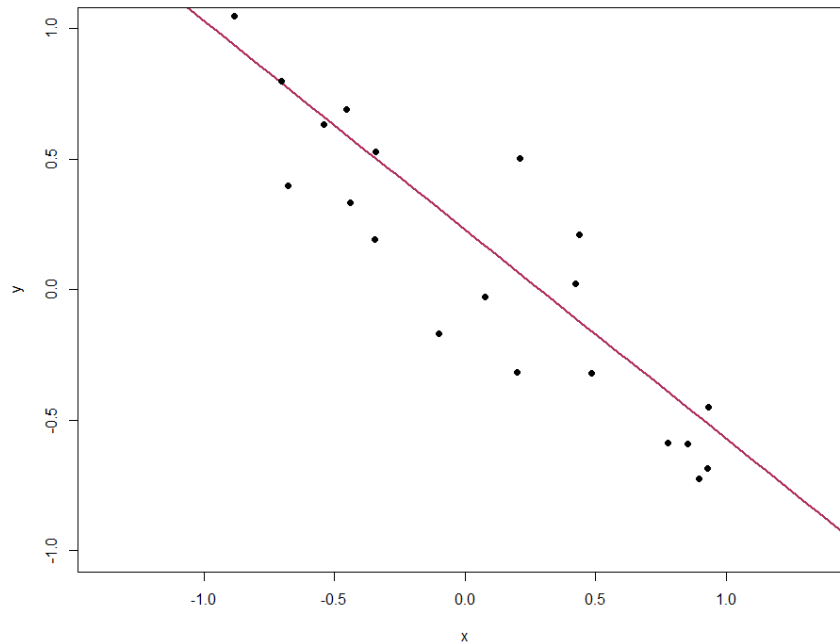
The choice of prior used will naturally have an influence on what the resulting posterior distribution looks like. The amount of data available will also play a role in shaping how much the posterior takes the form of the prior or of the likelihood. If we had used a non-informative prior, such as a zero-mean Gaussian with a very large variance, this prior would be contributing little new information to the problem (since it is essentially a constant) and hence the mean of the posterior distribution would coincide with the maximum likelihood estimator. Similarly, if we did not have any observed data points to work with, there would be no likelihood function and so the posterior distribution would simply be equal to the prior distribution (Bishop, 2006:153).

A simple simulation example can be used to demonstrate these different properties of Bayesian learning. We consider the  $p = 2$  dimensional scenario since this can be visualized graphically. Suppose that the true, underlying model is given by

$$y = \theta_0 + \theta_1 x = 0.23 - 0.8x \quad (4.21)$$

and data points are generated by adding an error term to points on this regression line. Hence, a random  $x$  value is generated between -1 and 1, the corresponding  $y$ -value is computed from Equation 4.21, and a zero-mean Gaussian random noise term,  $\varepsilon \sim \mathcal{N}(0, \tau^{-1})$  with  $\tau = 16$

(corresponding to a standard error of  $\frac{1}{4}$ ), is added to the result. This target function (in red) and 20 sample data points are depicted in Figure 4.1.



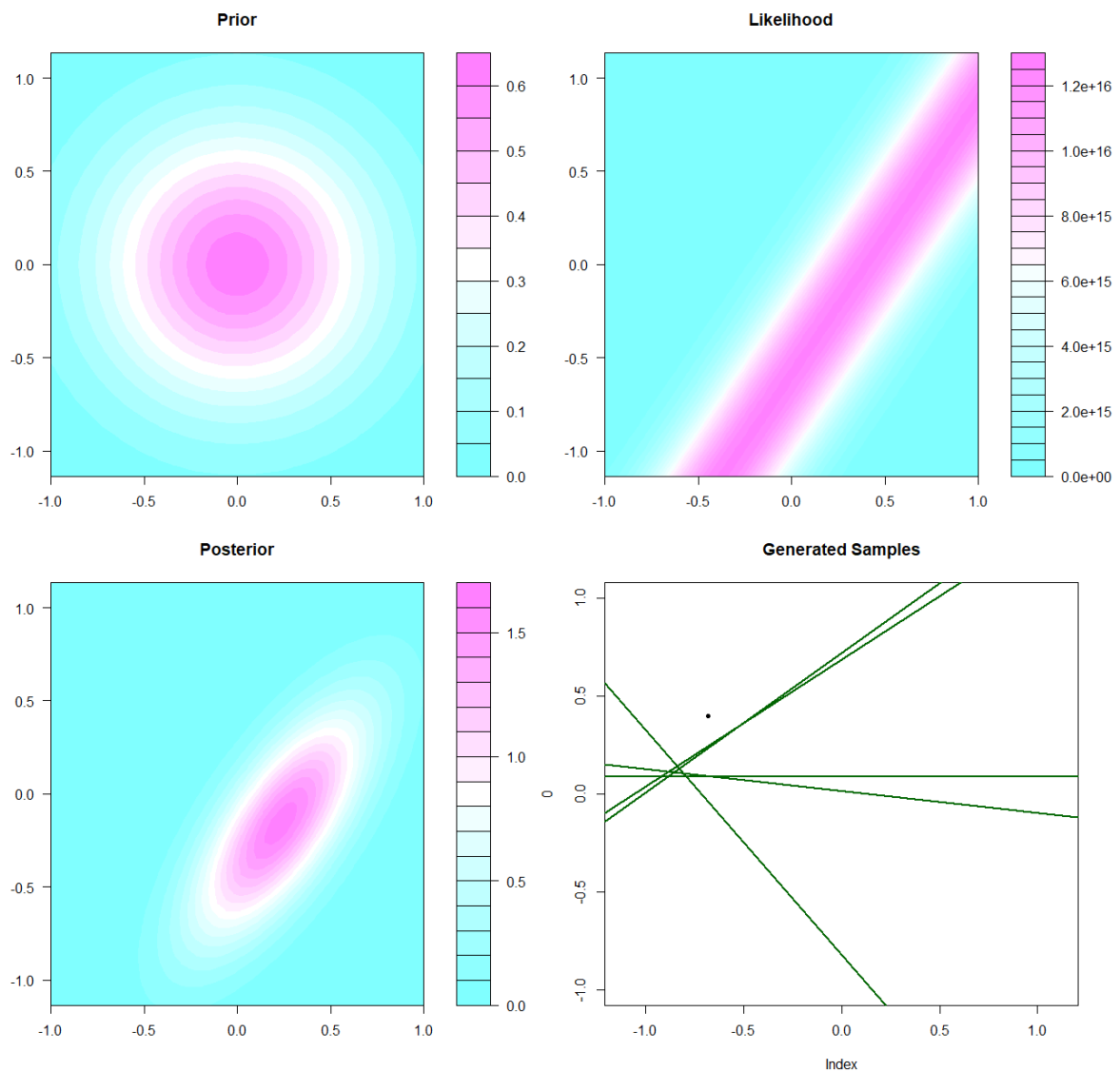
**Figure 4.1: Simulated data from the linear model.**

We will now use the Bayesian approach to linear regression to determine the posterior distribution for the parameters  $\theta_0$  and  $\theta_1$  and examine the effects of different prior distributions as well as the impact that sample size has on the resulting posterior distribution.

We will start by defining the prior distribution to be the zero-mean, white noise Gaussian prior given by

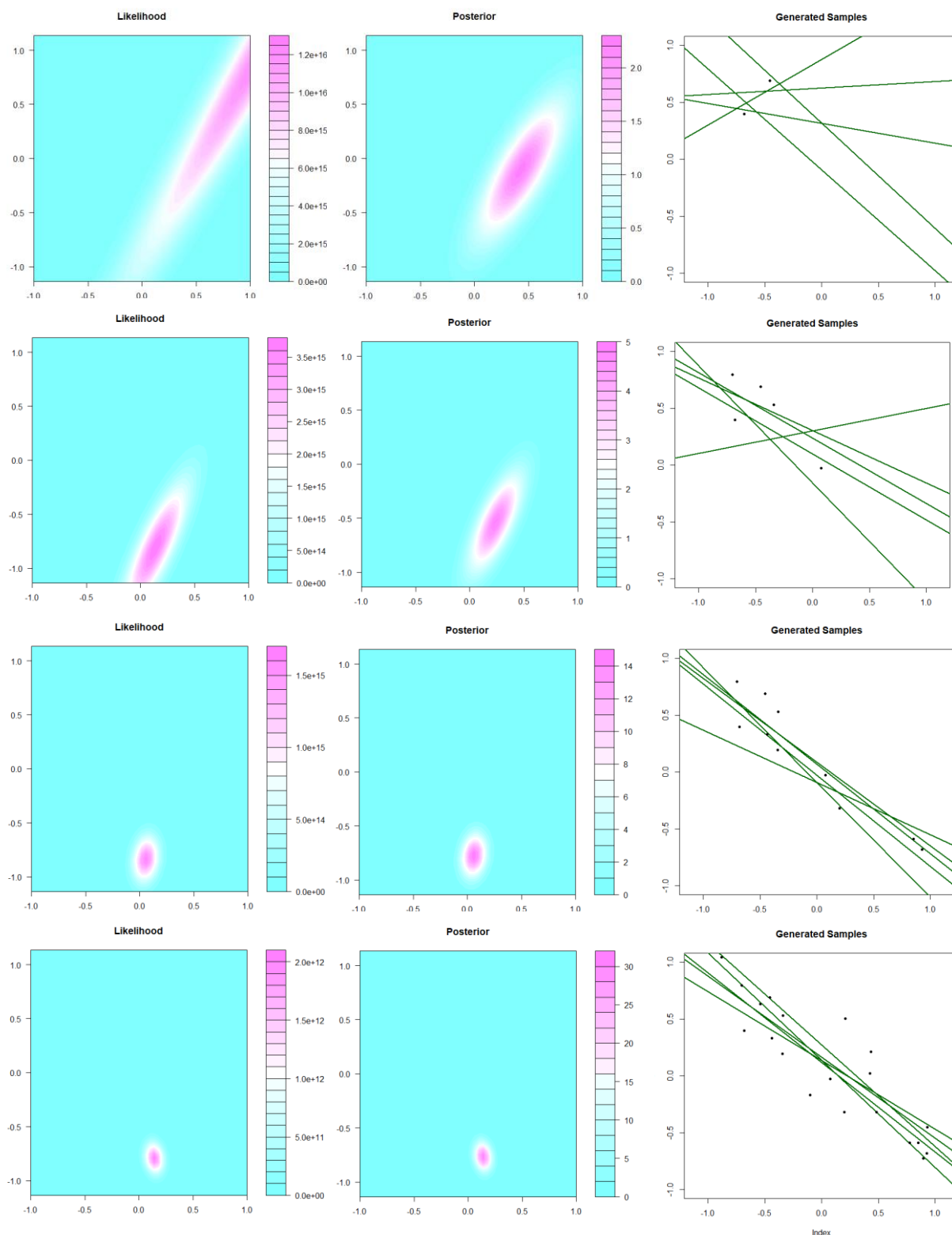
$$p(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \equiv \mathcal{N}\left(\mathbf{0}, \frac{1}{4}\mathbf{I}\right). \quad (4.22)$$

Firstly, consider the situation where the training sample contains only 1 observation. The prior, likelihood and posterior densities are displayed in Figure 4.2 below. The figure in the bottom right-hand corner denotes this single point as well as five different regression lines generated from the resulting posterior distribution. Since there was only one data point, these lines have a large variability.



**Figure 4.2: Bayesian regression using one data observation.**

We now sequentially add more observations into the observed data sample. Figure 4.3 denotes these resulting distributions and sampled regression lines from the posterior distribution. The first row corresponds to two points, the second row corresponds to five points, the third row to ten points and the final row denotes the full training set of twenty sample points.



**Figure 4.3: Bayesian regression with sequential inclusion of observed data points.**

This example clearly shows how the addition of more data points very quickly improves the accuracy of the Bayesian approach but also how well the Bayesian approach performs in small sample settings. The more training sample points that were included, the smaller the variance of the resulting posterior distribution. However, when only half of the training data was included, the

posterior distribution already becomes highly precise. The parameters of the posterior distribution after including all  $N = 20$  sample points are given by

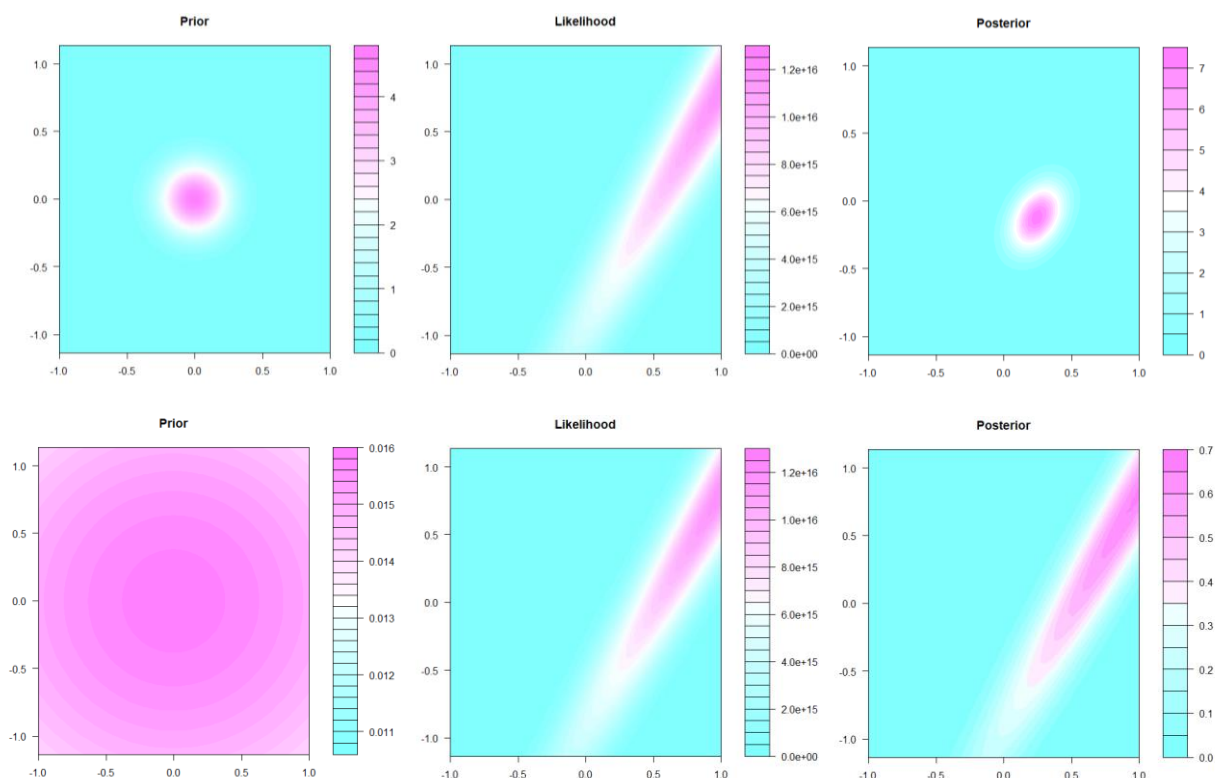
$$\boldsymbol{\mu}_N = [0.1391, -0.7630]^T \quad \text{and} \quad \boldsymbol{\Sigma}_N = \begin{bmatrix} 0.003149 & -0.000733 \\ -0.000733 & 0.008525 \end{bmatrix}. \quad (4.23)$$

Clearly, this mean vector is not too far off the true values of  $\boldsymbol{\theta} = [0.23, -0.8]^T$  if one were to consider the mean as a point estimate to summarize this distribution. This will be looked at further in the next section.

Also demonstrated in this example is how the addition of new observations update the likelihood function. Further, each time a new observation is included in the dataset, the previous posterior distribution becomes the new prior distribution used to obtain the posterior distribution after the new point has been included. Hence, posterior distributions are updated sequentially, each time becoming the prior distribution for the next iteration. This is one way to obtain the posterior distribution if data arrives individually. Alternatively, if the full dataset is available from the start, the same posterior distribution can be obtained by simply using the original prior distribution in Equation 4.22 and multiplying this by the likelihood of the full dataset.

The selected prior distribution in Equation 4.22 was only vaguely informative, and consequently had little effect on the final posterior distribution which took its form almost entirely from the likelihood function. We consider now two adjustments to the above simulation and consider firstly a very non-informative prior, followed by a highly informative prior. In both cases only 2 sample points were used to limit the relative influence of the likelihood function.

Since the posterior distribution is a product of the prior and the likelihood, it can be regarded as a trade-off between these two distributions. When there is little data available but a strong prior belief of the parameter values, the posterior distribution will take a form closer to that of the prior distribution since it will be providing more informative information into the model than the likelihood function. On the other hand, regardless of the size of the training set, if there is little to no relevant prior belief regarding the parameters and a flat/non-informative prior distribution is used, this prior will contribute virtually nothing to the result posterior and so the posterior distribution will mostly take the form of the likelihood function.



**Figure 4.4: Comparison of informative and non-informative priors.**

In all the previous derivations and applications, we worked with a conjugate prior so that the posterior distribution belonged to the same family of distributions as the prior distribution. The benefit of the Gaussian distribution is that it has a form that can be used as both an informative and an uninformative prior by simply adjusting the size of the variance. However, any appropriate prior distribution could have been used. As will be seen in Chapter 5, the use of some specific prior distributions can lead to variable selection in the model. None of these distributions are conjugate to a Gaussian distribution, however they are still very important prior distributions to be considered. The improvements of computing power have made the use of non-conjugate priors much more feasible because estimation methods are much quicker to implement when the posterior distribution cannot exist in closed form and is reliant on approximation.

#### 4.2.2 The MAP Estimator

The Bayesian approach to linear regression provides a distribution for the parameters of interest in contrast to classical regression which returns only a point estimate. Once the posterior has been obtained, the Bayesian approach does not further specify how this distribution should be summarized into a single statistic. This leads to the introduction of the maximum *a posteriori* (MAP) estimate that was briefly mentioned in Section 3.2.2.

The maximum *a posteriori* estimate is given by



$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}} p(\theta|\mathcal{D}). \quad (4.24)$$

In other words, the MAP estimates are those values of the parameters which maximize the posterior and can be found using any optimization method (Barber, 2011:184). In the example given previously, since the mode of a Gaussian coincides with the mean, the MAP estimate will simply be the posterior mean which was given by  $\mu_N = [0.1391, -0.7630]^T$ . Clearly, if we have a non-informative prior, such as the uniform distribution, then the MAP estimator will coincide with the maximum likelihood estimator, since  $p(\theta)$  will be a constant and hence not change for differing values of  $\theta$  (Barber, 2011:18). Further, in large samples, the likelihood function will almost always dominate anyway so even a highly informative prior might have very little influence on the posterior.

Despite the analytical appeal of the MAP estimator, it does have several drawbacks. The most notable being the fact that any point estimate does not provide the user with a measure of uncertainty and this is a very important quality of an estimate because one needs to have an idea of how far the estimate can be trusted. Not modelling the uncertainty can also lead to overfitting due to overconfidence. Furthermore, if the mode of the distribution happens to be an atypical point that does not adequately represent the distribution, the MAP estimator will be a poor choice since it will not satisfactorily summarize where the majority of the data points are actually occurring (Murphy, 2012:150).

Since the MAP estimator is only concerned with the mode of the posterior, normalization of the distribution was not necessary, i.e.  $p(y)$  was ignored. However, a fully Bayesian approach would require one to consider the posterior distribution in its entirety, taking  $p(y)$  into account to obtain the normalized posterior distribution. Evaluation of  $p(y)$  can often be very challenging and sometimes analytically intractable. Hence, approximation methods are required to estimate this marginal distribution, and these will be discussed later.

### 4.2.3 The Predictive Distribution

So far, we have described how to obtain the posterior distribution for the model parameters given some prior distribution. Continuing with this example, the posterior distribution was derived to be given by

$$p(\theta|y) \sim \mathcal{N}(\mu_N, \Sigma_N) \quad (4.25)$$

with  $\mu_N$  and  $\Sigma_N$  as defined previously in Equation 4.20.

We also showed how to summarize this distribution into a single point estimate, known as the MAP estimate. However, in practice, we are not typically interested in the different values of the parameter  $\theta$  itself, but rather in being able to make predictions for the response variable  $y$  when new data  $x$  arrives. This can be achieved by evaluating the predictive distribution, defined by,

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}; \tau) = \int p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{x}, \mathcal{D}; \tau) d\boldsymbol{\theta} = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}; \tau) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \quad (4.26)$$

where we have obtained the marginal distribution by simply integrating out the contribution of the parameter  $\boldsymbol{\theta}$  in the joint distribution. Given that  $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(\mathbf{x}^T \boldsymbol{\theta}; \tau^{-1})$  and  $p(\boldsymbol{\theta}|\mathcal{D}) \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$ , it follows that

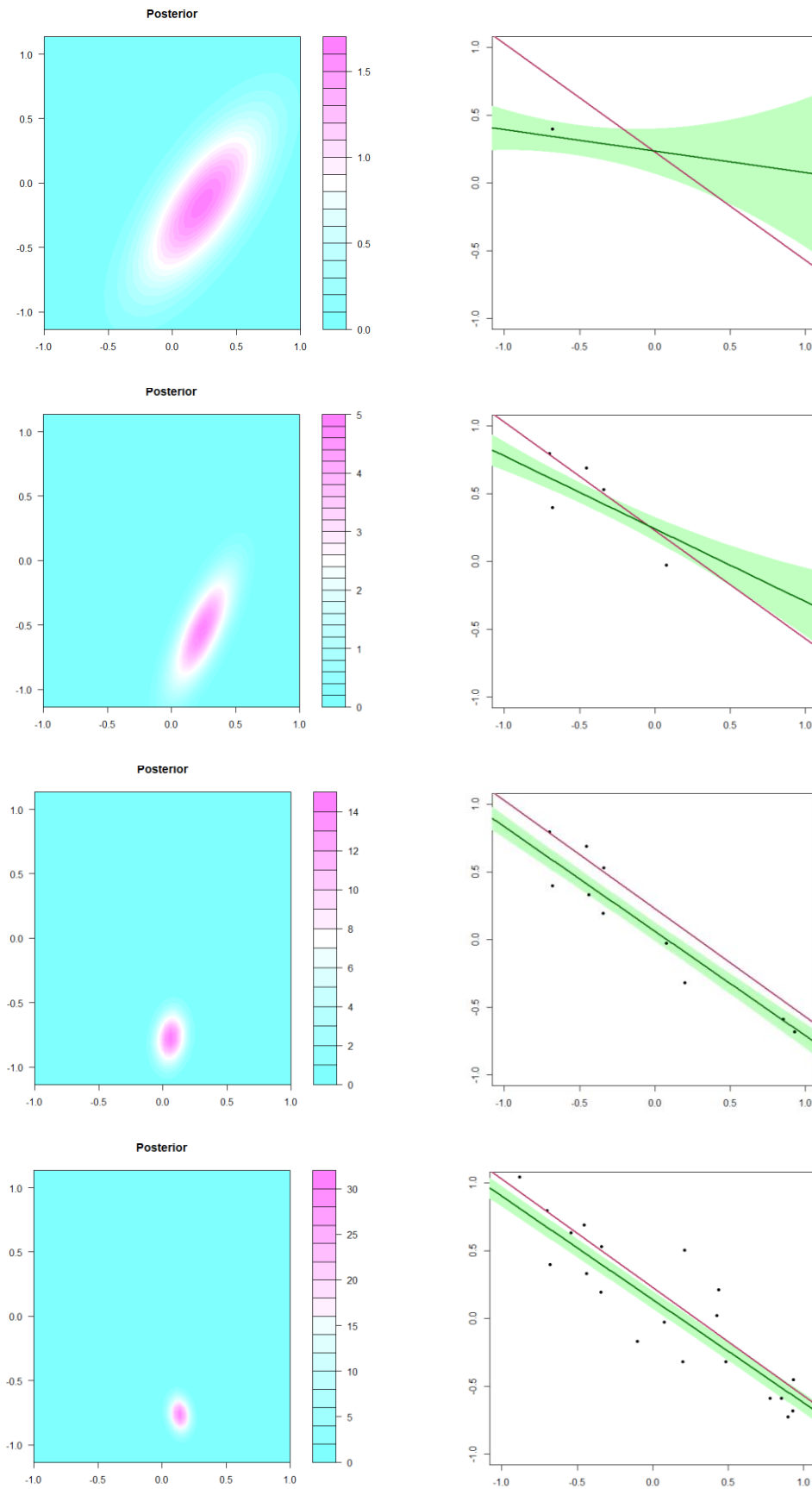
$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}; \tau) \sim \mathcal{N}(\boldsymbol{\mu}_N^T \mathbf{x}; \sigma_N^2) \quad (4.27)$$

where

$$\sigma_N^2 = \frac{1}{\tau} + \mathbf{x}^T \boldsymbol{\Sigma}_N \mathbf{x}. \quad (4.28)$$

This result follows in an analogous manner to the derivation done previously in Section 4.2 for the posterior distribution since we are again working with two Gaussian distributions.

Figure 4.5 on the following page displays the posterior distribution as well as the predictive distribution for the simulated dataset based on the same training samples of size  $N = 1, 5, 10$  and  $20$  as done previously. The red line shows the true underlying linear model from which the data was generated. The light-green shaded regions denote one-standard error on either side of the mean.



**Figure 4.5: Posterior and predictive distributions for the simulated data.**

Looking closer at the variance for the predictive distribution given in Equation 4.28, we see that the first term reflects the noise in the data, and the second term represents the associated uncertainty in the estimated parameters  $\theta$ . This leads to an interesting property that occurs as the sample size  $N$  increases. Table 4.1 shows the posterior predictive variance associated with increasing training sample sizes. The last column shows the difference between the posterior predictive variance and the noise parameter  $\tau = 16$ .

**Table 4.1: Changes in posterior predictive variance for increased sample size.**

Training sample size	$\sigma_N^2$	Difference
<b>1</b>	0.0718923839	0.0093923839
<b>2</b>	0.0676723103	0.0051723103
<b>5</b>	0.0646100390	0.0021100390
<b>10</b>	0.0639253291	0.0014253291
<b>20</b>	0.0634228101	0.0009228101
<b>50</b>	0.0628168110	0.0003168110
<b>100</b>	0.0626615565	0.0001615565
<b>1000</b>	0.0625176158	0.0000176158

If we consider the limit  $N \rightarrow \infty$ , then the variance of the posterior distribution tends towards zero due to  $\Sigma_N \rightarrow \mathbf{0}$ . Thus, the second term in the expression for the posterior predictive variance in Equation 4.28 will disappear, and so  $\sigma_N^2 \rightarrow \frac{1}{\tau}$  as  $N \rightarrow \infty$  as can be seen in Table 4.1. The value of the posterior predictive distribution can clearly be seen to be converging towards  $\frac{1}{\tau} = \frac{1}{16} = 0.0625$ . As a result, the only variance remaining in the predictive distribution comes from the noise inherent in the data. This noise obviously cannot be reduced any further and so this variance will be at a minimum (Bishop, 2006:156).

Hence, the Bayesian solution will coincide with the frequentist approach to regression when the datasets are very large since the posterior parameters will tend towards the maximum likelihood estimates. Some of the greatest advantages of the Bayesian approach over a frequentist approach are often seen in small samples settings where accurate estimation via point estimates tends to be poor. However, as with most modelling approaches, the results obtained via a Bayesian approach will continuously improve and become more accurate as the size of the available data increases and the Bayesian approach will always provide the advantage of

obtaining a full posterior distribution over the parameters. Bayesian approaches also allow for a more intuitive means of interpreting parameter values and performing inference due to having access to the full posterior distributions of the parameters rather than simply working with a point estimate.

### 4.3 THE FULLY BAYESIAN APPROACH

Up to this point, we have assumed that all the parameters for the distributions are known or can be determined. In practice, this is obviously not the case. Although it may be possible to determine the parameters of the posterior distribution experimentally, the parameters of the prior distribution still need to be specified. For example, in the derivation of the posterior distribution of the Gaussian done previously, we assumed that the noise precision parameter  $\tau$  was a known constant. Since this is typically not the case, we rather treat this as a hyperparameter to be estimated. It was also assumed that the functional form of the underlying model for this covariance matrix was known and given by a bivariate Gaussian distribution. Similarly, we assumed that we have a prior distribution with known parameters  $\mu_0$  and  $\Sigma_0$ .

In a fully Bayesian approach to inference, there are in fact three levels of inference that need to be performed since we need to consider (1) the functional form of the model, (2) the hyperparameters for the selected model, and (3) the model parameters. At the lowest level, there is the evaluation of the posterior over the parameters which is given by

$$p(\theta|X, y, \Gamma, \mathcal{M}_k) = \frac{p(y|X, \theta, \mathcal{M}_k)p(\theta|\Gamma, \mathcal{M}_k)}{p(y|X, \Gamma, \mathcal{M}_k)} \quad (4.29)$$

where  $\Gamma$  denotes the set of hyperparameters in the chosen model, and the chosen model is denoted by  $\mathcal{M}_k$ .

This is simply what was done in Equation 4.7; the dependence on the model choice and hyperparameter set was just suppressed to allow for uncluttered notation. The denominator in this expression – the normalizing constant – which is independent of the parameters  $\theta$ , is also known as the evidence function, and can be expressed as

$$p(y|X, \Gamma, \mathcal{M}_k) = \int p(y|X, \theta; \mathcal{M}_k)p(\theta|\Gamma, \mathcal{M}_k)d\theta. \quad (4.30)$$

The second level of estimation considers the hyperparameters. If we assign a hyper-prior,  $p(\Gamma|\mathcal{M}_k)$ , that is, a prior distribution for the hyperparameters, we can similarly obtain a posterior distribution for them as

$$p(\Gamma|y, X, \mathcal{M}_k) = \frac{p(y|X, \Gamma, \mathcal{M}_k)p(\Gamma|\mathcal{M}_k)}{p(y|X, \mathcal{M}_k)} \quad (4.31)$$

where the normalizing constant in this expression is given by

$$p(\mathbf{y}|\mathbf{X}; \mathcal{M}_k) = \int p(\mathbf{y}|\mathbf{X}; \boldsymbol{\Gamma}, \mathcal{M}_k) p(\boldsymbol{\Gamma}|\mathcal{M}_k) d\boldsymbol{\Gamma}. \quad (4.32)$$

Finally, the last step is to determine the posterior distribution for each of the proposed models, that is,

$$p(\mathcal{M}_k|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathcal{M}_k) p(\mathcal{M}_k)}{p(\mathbf{y}|\mathbf{X})} \quad (4.33)$$

where  $p(\mathbf{y}|\mathbf{X}) = \sum_{k=1}^K p(\mathbf{y}|\mathbf{X}, \mathcal{M}_k) p(\mathcal{M}_k)$  (Rasmussen & Williams, 2018:109).

Hence, to implement a fully Bayesian approach to linear regression requires the evaluation of several integrals, and often many of these will be analytically intractable or simply too complex to be worth evaluating exactly. Hence, efficient approximation methods have been developed to obtain estimates of these integrals to allow for faster and easier implementations of a fully Bayesian approach to linear regression problems.

#### 4.3.1 Bayesian Model Selection

Occam's razor is a philosophical principle stating that the simplest solution to a problem is most likely to be the correct one. Extending this to the statistical world translates to suggesting that models which make the fewest assumptions and/or contain the fewest parameters are typically the 'correct', or best, ones (MacKay, 2005: 343).

As done previously, the assumption is that the data can be modelled by the multiple linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon}. \quad (4.34)$$

However, this is clearly a very simple and straightforward model and alternatives to this could perhaps lead to a better fit. The previous theory in Section 4.2 can be extended directly to include any basis expansion of the parameters, with basis functions denoted by  $\phi_m(\mathbf{x})$ ,  $m = 1, \dots, M$ . In fact, the linear regression model given in Equation 4.34 is already in the form of a basis expansion whereby the basis functions are simply the identity function, that is,

$$\phi_m(\mathbf{x}_i) = x_{im}, \quad m = 1, \dots, p, i = 1, \dots, N. \quad (4.35)$$

Hence, the design matrix  $\mathbf{X}$  of predictors can be analogously defined by the set of basis functions which will be denoted by  $\boldsymbol{\Phi}: N \times (m + 1)$  where the  $ij$ th entry in  $\boldsymbol{\Phi}$  is given by

$$\Phi_{ij} = \phi_j(\mathbf{x}_i). \quad (4.36)$$

Hence, the multiple linear regression model in Equation 4.34 can equivalently be expressed as

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \quad (4.37)$$

and all the results derived in Sections 4.1 and 4.2 can immediately be obtained by simply replacing each design matrix  $X$  with the design matrix of basis functions  $\Phi$ . We then desire a method to be able to compare different choices of basis functions to determine which one results in the best fit without being too complex.

The basic idea behind Bayesian model selection is to assign a probability to each of the possible models to represent the level of uncertainty associated with that choice of model. Suppose that we are interested in comparing between a choice of  $K$  models, given by  $\mathcal{M}_k, k = 1, \dots, K$ . Each of these  $\mathcal{M}_k$  models refer to a specific probability distribution over the observed training data. An important aspect of Bayesian model selection is that the data is assumed to have been generated by one of the  $K$  models under consideration (i.e. one of the models represents the true model). However, we are uncertain about which one of these models is the true model, and this uncertainty can be expressed through the use of a prior probability, denoted by  $p(\mathcal{M}_k)$ . We can use this to express the fact that we may possibly believe that certain models are more likely than others. Alternatively, we could have no preference and assign equal prior probabilities to all models under consideration.

Given the training data,  $\mathcal{D}$ , we obtain a probability for comparison by determining the posterior distribution for each of the models, that is,

$$p(\mathcal{M}_k|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_k)p(\mathcal{M}_k). \quad (4.38)$$

The term  $p(\mathcal{D}|\mathcal{M}_k)$  is known as the model evidence, or marginal likelihood, and this term represents which model the data shows a preference towards, since it represents the probability of obtaining the observed data given that specific model choice.

In our scenario, the models include the unknown parameter vector  $\theta$ . Thus, the model evidence is given by,

$$p(\mathcal{D}|\mathcal{M}_k) = \int p(\mathcal{D}|\theta, \mathcal{M}_k)p(\theta|\mathcal{M}_k)d\theta \quad (4.39)$$

and the Bayesian model selection problem is equivalent to determining the model which maximizes this quantity. We note that this evidence function is exactly equal to the denominator of the posterior distribution for our parameter  $\theta$ . Including the dependence on the model into Bayes formula, we have

$$p(\theta|\mathcal{D}, \mathcal{M}_k) = \frac{p(\mathcal{D}|\theta, \mathcal{M}_k)p(\theta|\mathcal{M}_k)}{p(\mathcal{D}|\mathcal{M}_k)} \quad (4.40)$$

and so we see that the denominator here is indeed the evidence function given in Equation 4.39 above (Bishop, 2006:162; Murphy, 2012:156).

To determine which model fits the best among a set of competing models, we require a method of comparison. Two competing models can be compared directly using the Bayes factor. If we consider two models,  $\mathcal{M}_i$  and  $\mathcal{M}_j$ , then, using Equation 4.38, the ratios of their posterior distributions can be written as

$$\frac{p(\mathcal{M}_i|\mathcal{D})}{p(\mathcal{M}_j|\mathcal{D})} \propto \frac{p(\mathcal{D}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_j)p(\mathcal{M}_j)} \quad (4.41)$$

$$\Rightarrow \text{posterior odds} \propto \text{Bayes factor} \times \text{prior odds}.$$

Depending on the value of this Bayes factor, we can obtain an indication of which model there is more evidence to favour – if either. In the given example, since we are working with Gaussians, the evidence function can be evaluated directly. However, determining the model evidence for the Bayes factor tends to be very difficult and is typically approximated using MCMC.

Suppose in the example given in Section 4.2.1 we were uncertain as to what degree polynomial the data was generated from and we wished to compare polynomials of degree zero (i.e. a constant function) up to degree 10. As mentioned, all previous theory can be extended directly to include any number of basis functions in the model by simply replacing the design matrix  $X$  with the relevant design matrix of basis functions, denoted by  $\Phi$ . Hence, we know that the true model is given by

$$y = \theta_0 + \theta_1 x = 0.23 - 0.8x \quad (4.42)$$

which is a degree-1 polynomial. However, we may wish to model higher order polynomials using basis expansions. For example, a degree-4 polynomial can be obtained by assuming that the underlying model is given by

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4. \quad (4.43)$$

Hence, this model in Equation 4.43 has 5 basis functions (to include the intercept), each given by,

$$\phi_m(x) = x^m, m = 0, 1, \dots, 4. \quad (4.44)$$

The model evidence for a particular model  $\mathcal{M}_k$  can be evaluated as

$$\begin{aligned} p(\mathcal{D}|\mathcal{M}_k) &= \int p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_k) p(\boldsymbol{\theta}|\mathcal{M}_k) d\boldsymbol{\theta} \\ &= \int \left\{ \left( \frac{\tau}{2\pi} \right)^{\frac{N}{2}} \exp \left( -\frac{\tau}{2} (\mathbf{y} - \Phi \boldsymbol{\theta})^T (\mathbf{y} - \Phi \boldsymbol{\theta}) \right) \right\} \left\{ \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \exp \left( -\frac{\alpha}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} \right) \right\} d\boldsymbol{\theta} \\ &= \left( \frac{\tau}{2\pi} \right)^{\frac{N}{2}} \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \int \exp(-g(\boldsymbol{\theta})) d\boldsymbol{\theta} \end{aligned} \quad (4.45)$$

where



$$g(\boldsymbol{\theta}) = \frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{\alpha}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}. \quad (4.46)$$

Using a similar manipulation as done in Section 4.2, this exponent term can be rewritten as

$$\begin{aligned} & \frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{\alpha}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} \\ &= \frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N)^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N) + \frac{\alpha}{2}\boldsymbol{\mu}_N^T\boldsymbol{\mu}_N + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)^T\boldsymbol{\Sigma}_N^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_N) \end{aligned} \quad (4.47)$$

where

$$\boldsymbol{\Sigma}_N^{-1} = \alpha\mathbf{I} + \tau\boldsymbol{\Phi}^T\boldsymbol{\Phi} \quad \text{and} \quad \boldsymbol{\mu}_N = \tau\boldsymbol{\Sigma}_N\boldsymbol{\Phi}^T\mathbf{y}. \quad (4.48)$$

Thus,

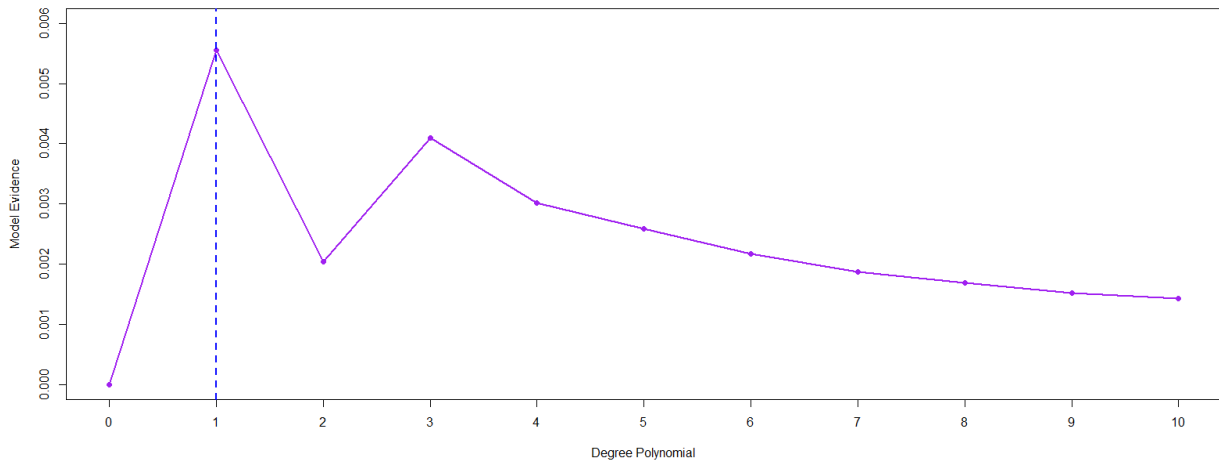
$$\begin{aligned} & \int \exp(-g(\boldsymbol{\theta})) d\boldsymbol{\theta} \\ &= \int \exp\left(-\frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N)^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N) - \frac{\alpha}{2}\boldsymbol{\mu}_N^T\boldsymbol{\mu}_N - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)^T\boldsymbol{\Sigma}_N^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)\right) d\boldsymbol{\theta} \\ &= \exp\left\{-\frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N)^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N) - \frac{\alpha}{2}\boldsymbol{\mu}_N^T\boldsymbol{\mu}_N\right\} \int \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)^T\boldsymbol{\Sigma}_N^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_N)\right) d\boldsymbol{\theta} \\ &= (2\pi)^{\frac{M}{2}}|\boldsymbol{\Sigma}_N^{-1}|^{-\frac{1}{2}} \exp\left\{-\frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N)^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N) - \frac{\alpha}{2}\boldsymbol{\mu}_N^T\boldsymbol{\mu}_N\right\} \end{aligned} \quad (4.49)$$

which follows from the normalizing constants of a Gaussian density function (Bishop, 2006:166–167).

Hence, we have

$$\begin{aligned} p(\mathcal{D}|\mathcal{M}_k) &= \left(\frac{\tau}{2\pi}\right)^{\frac{N}{2}} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \int \exp(-g(\boldsymbol{\theta})) d\boldsymbol{\theta} \\ &= \alpha^{\frac{M}{2}} \tau^{\frac{N}{2}} (2\pi)^{-\frac{N}{2}} |\boldsymbol{\Sigma}_N^{-1}|^{-\frac{1}{2}} \exp\left\{-\frac{\tau}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N)^T(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_N) - \frac{\alpha}{2}\boldsymbol{\mu}_N^T\boldsymbol{\mu}_N\right\}. \end{aligned} \quad (4.50)$$

The model evidence for each of these 11 polynomials is shown in Figure 4.6.



**Figure 4.6: Model evidence for different degree polynomials.**

Since the simulated data in Section 4.2.1 was generated from a linear model (a degree-1 polynomial), it is not surprising that the data showed the greatest favour towards this model. The next-best model based on this data is a polynomial of degree 3. We consider now the Bayes factor for comparison of these two models where no prior preference is allocated to either model. Denoting the  $k$ th degree polynomial by  $\mathcal{M}_k$ , the Bayes factor for these two models is given by

$$B_{13} = \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_3)} \approx 1.4. \quad (4.51)$$

Although this value is greater than 1, according to Kass and Raftery (1995), this value is “Not worth more than a bare mention” implying that the preference shown by the data towards the linear model is hardly greater than that shown to the degree-3 polynomial. This, of course, was only based off 20 sample points. If the sample was increased to 50, the bayes factor becomes

$$B_{13} = \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_3)} \approx 7.76 \quad (4.52)$$

which now shows substantial evidence towards the linear model. Comparison via Bayes factor, however, can also be subjective to the nature of the problem (Kass & Raftery, 1995).

The bias-variance trade-off was introduced in Section 1.2.1 and highlighted the issue of finding the best model to balance out the effects that bias and variance have on the test error. A common issue with model selection problems in general is that models with more parameters tend to be favoured over those with fewer parameters since more parameters means greater flexibility and a better fit. However, this often leads to heavily overfitted models. If we had performed the model selection problem using  $p(\mathcal{D}|\hat{\theta}^{MAP})$ , then we would consequently choose models with more parameters since they will have a higher likelihood from fitting the data better.

Common frequentist approaches for avoiding this issue is to make use of cross-validation methods whereby the model is fit several times to certain portions of the training data and then

tested on a held-out set of training data to evaluate its fit. However, this is only possible when the observed dataset is large enough to render this practical since a sufficient amount of data is required to build reasonable models. The benefit of the Bayesian approach to model selection is that overfitting is typically avoided by marginalizing over the model parameters instead of point estimates. If we consider again the form of the marginal likelihood,

$$p(\mathcal{D}|\mathcal{M}_k) = \int p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_k)p(\boldsymbol{\theta}|\mathcal{M}_k)d\boldsymbol{\theta} \quad (4.53)$$

we see that this marginalization can be regarded as averaging over different datasets that could arise from each model if we sample from the prior distribution for the parameters and then generatively obtain a dataset from the selected model based on these sampled parameter values. More complex models will generate a higher variability of datasets whereas simpler models, such as linear models, will generate datasets that look very similar to each other. This means that complex models will spread out the model evidence over a large range of datasets and thus not show strong evidence for any one particular dataset. On the other hand, very simple models will only generate a small variety of datasets and if these are not very similar to the true dataset, the marginal evidence will not show favour towards any of those models. Hence, marginalizing over the model parameters will result in the marginal likelihood favouring intermediate complexity and thus largely avoid overfitting (Bishop, 2006: 163). There is also no need for a validation set so the full training dataset can be used which is especially beneficial when data is scarce. Also, in contrast to cross-validation methods, multiple runs of the training data over different models is not needed which subsequently reduces the required training time.

#### 4.3.2 Type-II Maximum Likelihood (ML-II)

The posterior distribution for the hyperparameters for the selected model,  $\mathcal{M}_k$ , is given by

$$p(\boldsymbol{\Gamma}|\mathbf{y}, \mathbf{X}, \mathcal{M}_k) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\Gamma}, \mathcal{M}_k)p(\boldsymbol{\Gamma}|\mathcal{M}_k)}{p(\mathbf{y}|\mathbf{X}, \mathcal{M}_k)} \quad (4.54)$$

which can be expressed more simply, but equivalently, as

$$p(\boldsymbol{\Gamma}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\Gamma})p(\boldsymbol{\Gamma}). \quad (4.55)$$

This step of inference tends to be very difficult to compute analytically and so is usually avoided by rather obtaining estimates of the hyperparameters using another method. One possible way to obtain a single value to summarize this distribution is by using a MAP estimate, that is, the hyperparameter set that maximizes the posterior distribution,

$$\boldsymbol{\Gamma}^{MAP} = \underset{\boldsymbol{\Gamma}}{\operatorname{argmax}} p(\boldsymbol{\Gamma}|\mathcal{D}) = \underset{\boldsymbol{\Gamma}}{\operatorname{argmax}} p(\mathcal{D}|\boldsymbol{\Gamma})p(\boldsymbol{\Gamma}). \quad (4.56)$$

If we have weak prior information about the hyperparameters so that  $p(\boldsymbol{\Gamma}) \approx c$ , for some constant  $c$ , then the maximization reduces to,

$$\mathbf{\Gamma}^{MAP} \approx \underset{\mathbf{\Gamma}}{\operatorname{argmax}} p(\mathcal{D}|\mathbf{\Gamma}) \quad (4.57)$$

and this is exactly equal to determining the set of hyperparameters which maximizes the marginal likelihood, since,

$$p(\mathcal{D}|\mathbf{\Gamma}) = \int p(\mathcal{D}|\mathbf{\Gamma}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{\Gamma}) d\boldsymbol{\theta}. \quad (4.58)$$

This method of fixing the hyperparameter set is known as the evidence procedure, or ML-II, where ‘ML’ stands for the maximum likelihood estimation step, and the ‘II’ represents that we are working at the second level, namely, the hyperparameters as opposed to the model parameters.

In some cases, evaluation of this integral is possible to do numerically, however, numerical integration is not always possible, or will be too complex to perform, and so approximation methods are needed and often preferred.

#### 4.3.2.1 The EM Algorithm

When numerical integration of the evidence function is not possible or feasible to perform, the expectation-maximization (EM) algorithm can be used to find estimates for the hyperparameters. In this algorithm, the model parameters  $\boldsymbol{\theta}$  are treated as latent variables and the values of the hyperparameters are estimated iteratively until convergence of the algorithm. The EM algorithm in general comprises three main tasks: data augmentation, expectation, and maximization.

In the first step, data augmentation, the space of observed sample points is enlarged by including latent (unobserved) data into the model. Then, to begin the iterative algorithm, initial values are assigned to the parameters of interest which need to be estimated. The iteration then begins, alternating between the expectation step and the maximization step until convergence. The expectation step sets up a probability distribution based on the probabilities associated with the current parameter estimates. The maximization step then uses this distribution to update the parameter estimates.

The reason for the success of the EM algorithm is that it can be shown that this iterative procedure will never decrease the log-likelihood as the process evolves. It is not, however, guaranteed to find the best estimates each time since it may converge to local rather than global maxima, but this can easily be improved by using several different starting values for the parameters (Hastie *et al.*, 2009:276).

Suppose that we denote the observed data by  $\mathcal{X}$  and the set of latent (unobserved) data by  $\mathcal{X}^L$ . The complete, or augmented, dataset is then given by  $\mathcal{X}^* = \{\mathcal{X}, \mathcal{X}^L\}$ . Further, let the distribution function of the augmented data be parameterized by the unknown parameters,  $\mathbf{\Gamma}$ . Since we have latent data, the complete log-likelihood function is given by  $p(\mathcal{X}, \mathcal{X}^L; \mathbf{\Gamma})$  and typical maximum-likelihood estimation clearly cannot be performed. The EM algorithm is summarized below.

### The Expectation-Maximization Algorithm

1. Assign starting values to the parameters,  $\Gamma^{(0)}$
2. Iterate:
  - a. **Expectation step:** at the  $j$ th step, determine  $p(\mathcal{X}^L|\mathcal{X}, \Gamma^{(j)})$  and compute the expectation

$$Q(\Gamma, \Gamma^{(j)}) = E[\log p(\mathcal{X}, \mathcal{X}^L; \Gamma)]$$

with the expectation taken with respect to  $p(\mathcal{X}^L|\mathcal{X}; \Gamma^{(j)})$

- b. **Maximization step:** Compute the updated parameter estimates,  $\Gamma^{(j+1)}$ , by finding

$$\Gamma^{(j+1)} = \underset{\Gamma}{\operatorname{argmax}} Q(\Gamma, \Gamma^{(j)})$$

- c. Check for convergence, otherwise return to step 2a.

The reason that the steps in the EM algorithm are possible to compute is because the posterior distribution  $p(\mathcal{X}^L|\mathcal{X}; \Gamma)$  is known, provided that the parameters  $\Gamma$  are known. This is how the iterations of the EM algorithm proceed, by beginning with some arbitrary value for  $\Gamma$ . As mentioned, the success of the EM algorithm is owed to the fact that the log-likelihood function will never decrease as the algorithm evolves.

Returning to the regression scenario and to simplify explanations further, we consider a zero-mean white noise prior for  $\theta$  that is dependent on the unknown precision parameter  $\alpha$ , that is,

$$p(\theta|\alpha) \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I}). \quad (4.59)$$

This means that the resulting posterior distribution will still be Gaussian, with mean and variance now given by

$$\Sigma_N^{-1} = \alpha\mathbf{I} + \tau\mathbf{X}^T\mathbf{X} \quad (4.60)$$

$$\mu_N = \tau\Sigma_N\mathbf{X}^T\mathbf{y} \quad (4.61)$$

following directly from the results given in Equation 4.20.

Using this, the hyperparameter set is given by  $\Gamma = \{\alpha, \tau\}$  which we will denote by the vector  $\Gamma = [\alpha, \tau]^T$ . The goal of the EM algorithm is to estimate values for these hyperparameters from the data itself by maximizing the evidence function. As mentioned, the EM algorithm requires a set of observed data and a set of latent data. Hence, we make use of the observed dataset  $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$  and treat the parameter set  $\theta$  as the latent data in the algorithm.

The EM algorithm begins by assigning initial values to the hyperparameters,  $\Gamma^{(0)} = [\alpha^{(0)}, \tau^{(0)}]^T$ . We then iterate between the expectation and maximization steps. In the expectation step of the

$j$ th iteration of the algorithm, we compute the posterior distribution  $p(\boldsymbol{\theta}|\mathbf{y}; \boldsymbol{\Gamma}^{(j)})$  which, as given previously in Equations 4.60 and 4.61, is fully specified with mean and covariance given by

$$\boldsymbol{\Sigma}_N^{(j)} = (\alpha^{(j)} \mathbf{I} + \tau^{(j)} \mathbf{X}^T \mathbf{X})^{-1} \quad (4.62)$$

$$\boldsymbol{\mu}_N^{(j)} = \tau^{(j)} \boldsymbol{\Sigma}_N^{(j)} \mathbf{X}^T \mathbf{y}. \quad (4.63)$$

We then need to determine the expected value of the log-likelihood function associated with the augmented dataset,

$$\log p(\mathbf{y}, \boldsymbol{\theta}; \boldsymbol{\Gamma}) = \log p(\mathbf{y}, \boldsymbol{\theta}; \alpha, \tau) = \log p(\mathbf{y}|\boldsymbol{\theta}; \tau) p(\boldsymbol{\theta}; \alpha) \quad (4.64)$$

which is given by

$$\log p(\mathbf{y}, \boldsymbol{\theta}; \boldsymbol{\Gamma}) = \frac{N}{2} \log \tau + \frac{p}{2} \log \alpha - \left( \frac{N}{2} + \frac{p}{2} \right) \log 2\pi - \frac{\tau}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) - \frac{\alpha}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}. \quad (4.65)$$

We now require the expected value of this quantity with respect to the latent variable  $\boldsymbol{\theta}$ , that is,

$$\begin{aligned} E_{\boldsymbol{\theta}}[\log p(\mathbf{y}, \boldsymbol{\theta}; \boldsymbol{\Gamma})] &= \frac{N}{2} \log \tau + \frac{p}{2} \log \alpha - \left( \frac{N}{2} + \frac{p}{2} \right) \log 2\pi \\ &\quad - \frac{\tau}{2} E_{\boldsymbol{\theta}}[(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})] - \frac{\alpha}{2} E_{\boldsymbol{\theta}}[\boldsymbol{\theta}^T \boldsymbol{\theta}]. \end{aligned} \quad (4.66)$$

Considering each expectation above in turn, we firstly have

$$\mathbf{A} := E_{\boldsymbol{\theta}}[\boldsymbol{\theta}^T \boldsymbol{\theta}] = \text{trace}(\boldsymbol{\Sigma}_N^{(j)}) + \boldsymbol{\mu}_N^{(j)T} \boldsymbol{\mu}_N^{(j)}. \quad (4.67)$$

And, using this, we have

$$\begin{aligned} \mathbf{B} &:= E_{\boldsymbol{\theta}}[(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})] \\ &= E_{\boldsymbol{\theta}}[\mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X}\boldsymbol{\theta}] \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X} \boldsymbol{\mu}_N^{(j)} + \text{trace}(\mathbf{X} \boldsymbol{\Sigma}_N^{(j)} \mathbf{X}^T) + \boldsymbol{\mu}_N^{(j)T} \mathbf{X}^T \mathbf{X} \boldsymbol{\mu}_N^{(j)} \\ &= (\mathbf{y} - \mathbf{X} \boldsymbol{\mu}_N^{(j)})^T (\mathbf{y} - \mathbf{X} \boldsymbol{\mu}_N^{(j)}) + \text{trace}(\mathbf{X} \boldsymbol{\Sigma}_N^{(j)} \mathbf{X}^T). \end{aligned} \quad (4.68)$$

Thus,

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}^{(j)}) &= E_{\boldsymbol{\theta}}[\log p(\mathbf{y}, \boldsymbol{\theta}; \boldsymbol{\Gamma})] \\ &= \frac{N}{2} \log \tau + \frac{p}{2} \log \alpha - \left( \frac{N}{2} + \frac{p}{2} \right) \log 2\pi - \frac{\tau}{2} \mathbf{B} - \frac{\alpha}{2} \mathbf{A}. \end{aligned} \quad (4.69)$$

Lastly, the maximization step then maximizes this quantity with respect to each of the hyperparameters to determine the new parameter updates. Hence,

$$\alpha^{(j+1)}: \frac{\partial}{\partial \alpha} \mathcal{Q}(\alpha, \tau, \alpha^{(j)}, \tau^{(j)}) = 0$$

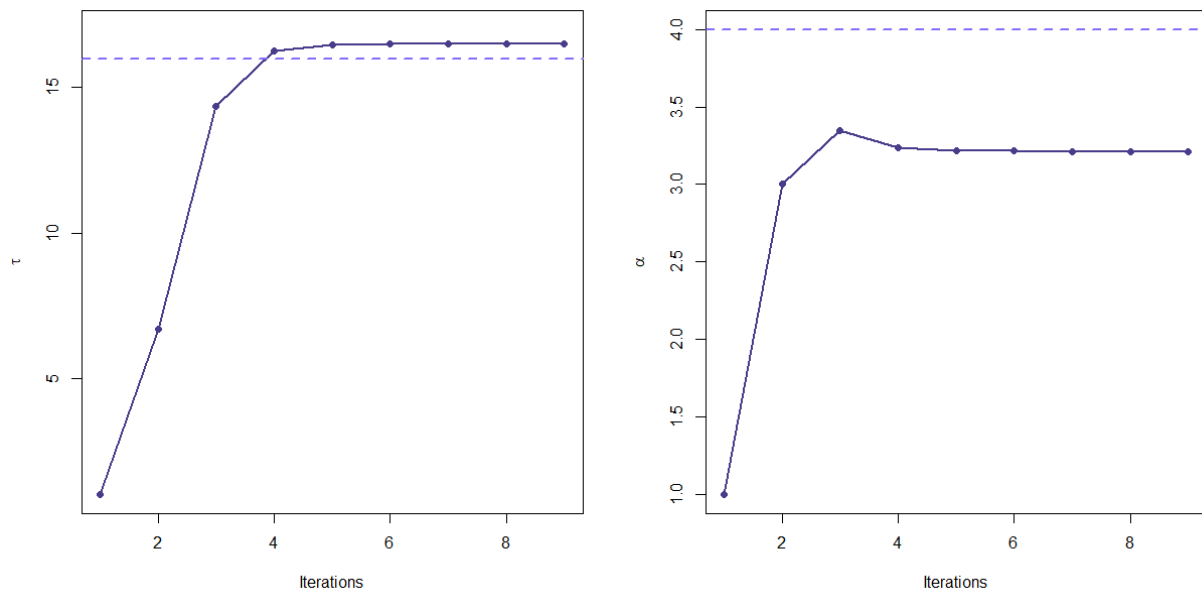
$$\begin{aligned} \Rightarrow \frac{p}{2\alpha^{(j+1)}} - \frac{1}{2}\mathbf{A} &= 0 \\ \Rightarrow \alpha^{(j+1)} &= \frac{p}{\text{trace}(\mathbf{\Sigma}_N^{(j)}) + \mathbf{\mu}_N^{(j)T} \mathbf{\mu}_N^{(j)}} \end{aligned} \quad (4.70)$$

and

$$\begin{aligned} \tau^{(j+1)}: \frac{\partial}{\partial \tau} Q(\alpha, \tau, \alpha^{(j)}, \tau^{(j)}) &= 0 \\ \Rightarrow \frac{N}{2\tau^{(j+1)}} - \frac{1}{2}\mathbf{B} &= 0 \\ \Rightarrow \tau^{(j+1)} &= \frac{N}{(\mathbf{y} - \mathbf{X}\mathbf{\mu}_N^{(j)})^T (\mathbf{y} - \mathbf{X}\mathbf{\mu}_N^{(j)}) + \text{trace}(\mathbf{X}\mathbf{\Sigma}_N^{(j)}\mathbf{X}^T)}. \end{aligned} \quad (4.71)$$

This process will continue until either the difference in consecutive parameter estimates is smaller than some pre-specified value or until the maximum number of iterations is reached (Theodoridis, 2015:606). Hence, once the best model has been chosen and the hyperparameters have been tuned to their optimal values, the posterior distribution of the model parameters can be found as well as the resulting predictive distribution.

Since it was verified that the linear model was the preferred model of choice, we can use the EM algorithm to tune the hyperparameters  $\alpha$  and  $\tau$ . Figure 4.7 shows the values of these hyperparameters after each iteration. The algorithm converged after only 9 iterations.



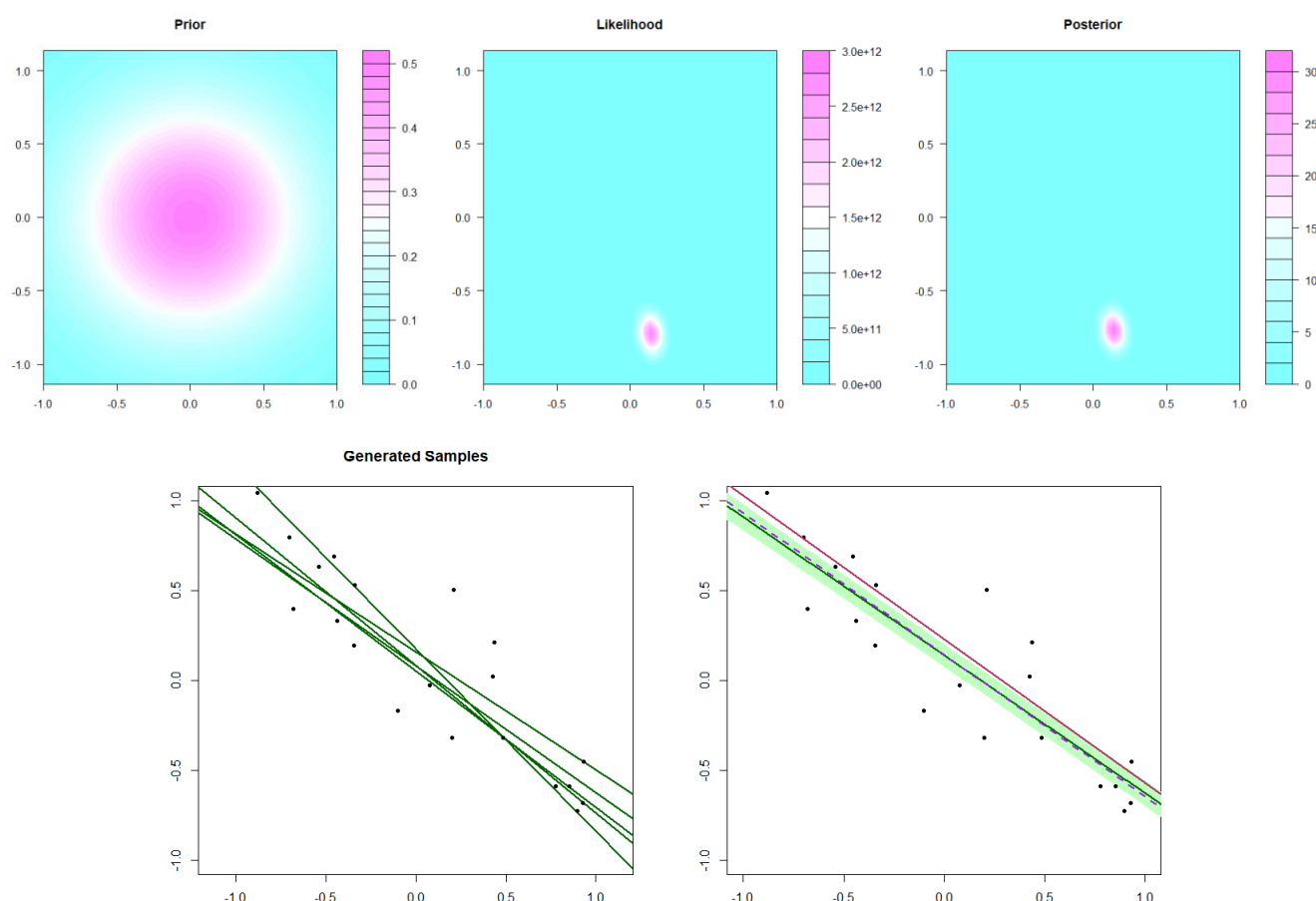
**Figure 4.7: Results of the EM algorithm.**

The converged values for  $\alpha$  and  $\tau$  are

$$\alpha = 3.214333 \quad \text{and} \quad \tau = 16.505722. \quad (4.72)$$

Given that the true value of  $\tau$  was 16, we can see that the EM algorithm did well to recover this. Further, the assumed value of  $\alpha = 4$  was not far off this empirically obtained value.

Hence, a fully Bayesian approach to this problem would have started with the model selection, whereby the linear model would have been chosen. Secondly, the EM algorithm would have been executed to obtain the values of  $\alpha$  and  $\tau$  to be used in the analysis. Finally, using this information, the posterior distribution and predictive distribution would be obtained. Using this fully Bayesian approach and the  $N = 20$  sample points, the prior, likelihood and posterior distribution are shown in Figure 4.8 followed by the posterior predictive distribution in the bottom right-hand corner. Since the chosen value of  $\alpha$  was close to the value obtained in the EM algorithm, the results are very similar to those obtained previously.



**Figure 4.8: Fully Bayesian approach to the linear regression problem.**

Also included in the plot for the predictive distribution is the regression line (dashed purple) which would have been obtained via a frequentist approach to this regression problem. Although the lines are almost identical, the immediate benefit of the Bayesian approach is apparent by the direct incorporation of the errors associated with the parameter estimates as well as the good performance when only very few data points were observed.



#### 4.4 SUMMARY

Bayesian approaches to linear regression perform well in small sample settings and do not require very large datasets to become reliable. Even though a fully Bayesian approach requires the evaluation of several different (and possibly very complex) integrals, efficient algorithms make approximations of these integrals timeous to execute. As the size of the available training data increases, the posterior predictive variance tends towards a minimum and the results of the Bayesian analysis will coincide with the maximum likelihood estimators.

Bayesian model comparison via the Bayes factor allows for direct comparison of different models based only on the training data. Unlike cross-validation, Bayesian model comparison does not require multiple iterations over the training data to determine which model is best suited. Since the models are compared based on their marginal likelihoods, the data will show preference towards models of intermediate complexity and be less prone to overfitting than if only a single point estimate was used. The advantage of a fully Bayesian approach to linear regression is apparent to see, particularly in the situation where training data may be sparse, since we obtain a full posterior distribution over the parameter estimates which results in the uncertainty of the estimate being immediately incorporated into the results.

## CHAPTER 5

### BAYESIAN VARIABLE SELECTION

#### 5.1 THE VARIABLE SELECTION PROBLEM

In the linear regression model with  $p$  potential parameters, there are a total of  $2^p$  possible models to choose from if we are to consider all combinations of including/excluding each variable from the model. The variable selection problem in general is concerned with finding efficient algorithms to determine which of these  $2^p$  models is the best to choose – the efficiency being crucial since  $2^p$  grows quickly for large  $p$ . Common frequentist approaches to variable selection include subset selection and regularization, among others. This chapter considers different Bayesian approaches to the variable selection problem. Variable selection is an extremely important step in any analysis since it can have a huge impact on the efficiency and prediction accuracy of the final model.

The choice of the prior distribution in a model has a large influence on the resulting posterior distribution of the parameters. In Chapter 4 on linear regression, we worked with a conjugate prior which meant that we used a prior distribution that was conjugate to the likelihood function, leading to a posterior distribution belonging to the same family of distributions as the prior. This is a very useful result that often makes determining the posterior distribution more convenient. However, as discussed in Chapter 2, there are many different priors which can be assigned that are not necessarily conjugate, or even proper density functions.

The prior distribution can also provide another useful advantage during inference, namely, variable selection. This chapter will consider two different priors that force some of the parameter estimates to zero, thereby inducing sparsity into the model. These priors are Zellner's  $g$ -prior and the spike-and-slab prior. Following from that a Bayesian perspective of ridge regression and the LASSO (Least Absolute Shrinkage and Selection Operator) will be discussed. Finally, although not strictly a variable selection technique, relevance vector machines for regression and classification are described as a technique that rivals that of the support vector machine. Relevance vector machines are included in this variable selection chapter because it is a sparse Bayesian learning technique that relies only on a very small subset of basis functions in the model by setting the coefficients of the remaining basis functions to zero, thus removing them from the model. This approach is similar in construction to the support vector machine but tends to result in a much sparser fit.

## 5.2 ZELLNER'S G-PRIORS

As mentioned, the choice of prior distribution can have a significant impact on the resulting posterior distribution and certain choices of priors can lead to variable selection. We consider again the multiple linear regression model

$$Y = X\theta + \varepsilon \quad (5.1)$$

where  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \tau^{-1}I)$ . The Bayesian approach to linear regression aims to determine the posterior distribution of the unknown parameter coefficients  $\theta$  by assigning a prior distribution to each of these unknown parameters in the model. The variable selection problem is concerned with determining a subset of the original predictor variables that will reduce the complexity of the model without comprising significantly on prediction error. In Section 4.3.1, the Bayesian model selection problem was introduced. In that section, we saw that two competing models, say  $\mathcal{M}_j$  and  $\mathcal{M}_k$ , can be compared by considering their Bayes factor

$$B_{jk} = \frac{p(\mathcal{D}|\mathcal{M}_j)}{p(\mathcal{D}|\mathcal{M}_k)} \quad (5.2)$$

i.e. the ratio of the preferences that the data shows to each of the competing models. In this Bayesian approach to model selection we require the specification of a prior distribution for the model parameters to determine the marginal likelihood of the data based on a particular model. Since prior distributions can have a large influence on the resulting posterior distribution, particular forms of priors that lead to variable selection have been proposed. Zellner's  $g$ -prior (Zellner, cited in Liang *et al.*, 2008) is an example of a prior distribution that leads to variable selection.

To understand the motivation behind Zellner's  $g$ -prior, we firstly consider the concept of the Fisher information. Informally, the Fisher information, denoted by  $\mathcal{I}(\cdot)$ , gives an indication of the amount of information that a dataset provides about some unknown parameter. In the specific case of multiple linear regression, the inverse Fisher information matrix is also equal to the covariance matrix of the maximum likelihood estimator  $\hat{\theta}$  obtained using ordinary least squares, that is,

$$\begin{aligned} \mathcal{I}(\theta) &= -E_y \left[ \frac{\partial^2}{\partial \theta \partial \theta^T} \log p(y|X, \theta; \tau) \right] \\ &= -E_y \left[ \frac{\partial^2}{\partial \theta \partial \theta^T} \left\{ \frac{N}{2} \log \tau - \frac{N}{2} \log 2\pi - \frac{\tau}{2} (y - X\theta)^T (y - X\theta) \right\} \right] \\ &= -E_y \left[ -\frac{\tau}{2} \cdot 2X^T X \right] \\ &= \tau X^T X. \end{aligned} \quad (5.3)$$

Hence, we see that the covariance matrix of  $\hat{\theta}$  is exactly equal to the inverse of the Fisher information matrix. As mentioned, this Fisher information matrix provides a means of measuring

how much information the data gives us about the parameter set  $\theta$ . Larger values of Fisher information represent less uncertainty due to a smaller variance around the MLEs in the likelihood function. This means that the inverse of the Fisher information will be smaller when there is more information available in the data for certain parameters.

This then provides the motivation for the use of Zellner's  $g$ -prior, which is given by

$$p(\theta|\tau, X) \sim \mathcal{N}\left(\theta_0, \frac{g}{\tau}(X^T X)^{-1}\right) \quad (5.4)$$

where  $g$  is a known parameter. Thus, we see that Zellner's  $g$ -prior is a data-dependent prior. Specific choices of how to obtain the value of  $g$  will be considered later in this section. The suggestion was to set the prior mean  $\theta_0$  equal to the values of  $\theta$  that are to be tested, i.e. for variable selection we set  $\theta_0 = \mathbf{0}$  since this choice shrinks the coefficients towards zero. The covariance matrix of Zellner's  $g$ -prior is exactly equal to the inverse Fisher information matrix, multiplied by the value of  $g$  (Nielsen *et al.*, 2014).

The intuition behind the use of this prior is that the parameters for which the data carries a large volume of information will be dominated by the likelihood function since the data should be representing less uncertainty around these estimates, and therefore the prior distribution, which is suggesting parameter values close to zero, will have little effect due to having a larger variability around those points. In other words, a higher Fisher information means a smaller value for the inverse Fisher information which corresponds to a flatter prior distribution with higher variability around the parameters for which that data has more information – i.e. the parameters would remain included in the model. Conversely, those parameters which should technically be removed from the model should have more variable information about them contained in the data, and this will be reflected via smaller values of the likelihood function and smaller Fisher information, allowing for the prior distribution to have a greater impact due to a smaller variance. As a result of this, those parameters which should be removed from the model will have posterior values clustered around zero. Clearly, the choice of  $g$  and  $\tau$  will also influence the relative role that the prior and the likelihood will in contributing to the posterior.

For this, Jeffrey's prior is typically assigned to  $\tau$  which is a non-informative prior and is defined by the inverse of the error variance. Hence, Jeffrey's prior for the precision parameter  $\tau$  is given by

$$p(\tau) \propto \frac{1}{\tau}. \quad (5.5)$$

The inclusion of Jeffrey's prior was not part of the original proposal of the  $g$ -prior but has now become the widely accepted standard formulation. The motivation behind this prior is that it is a non-informative prior that is invariant to scale and location transformations, whereas, for example, the uniform distribution is not (Liang *et al.*, 2008).

Using this definition for Zellner's  $g$ -prior, with  $\theta_0 = \mathbf{0}$ , and assuming without loss of generality that the data are centered to absorb any intercept term, the full posterior distribution is given by

$$\begin{aligned}
 p(\theta, \tau | \mathbf{y}, \mathbf{X}) &\propto p(\mathbf{y} | \theta, \tau, \mathbf{X}) p(\theta | \tau, \mathbf{X}) p(\tau) \\
 &= \left\{ \tau^{\frac{N}{2}} (2\pi)^{-\frac{N}{2}} \exp \left( -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \right) \right\} \cdot \left\{ \frac{(2\pi)^{-\frac{p}{2}}}{\left| \frac{g}{\tau} (\mathbf{X}^T \mathbf{X})^{-1} \right|^{\frac{1}{2}}} \exp \left( -\frac{1}{2} \theta^T \left[ \frac{g}{\tau} (\mathbf{X}^T \mathbf{X})^{-1} \right]^{-1} \theta \right) \right\} \cdot \frac{1}{\tau} \\
 &\propto \left\{ \tau^{\frac{N}{2}-1} \exp \left( -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \right) \right\} \cdot \left\{ \tau^{\frac{p}{2}} \exp \left( -\frac{\tau}{2g} \theta^T \mathbf{X}^T \mathbf{X} \theta \right) \right\} \\
 &= \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \right\} \exp \left\{ -\frac{\tau}{2g} \theta^T \mathbf{X}^T \mathbf{X} \theta \right\}. \tag{5.6}
 \end{aligned}$$

We consider now the first exponential term above and rewrite the quadratic as two separate quadratics in terms of the MLE from the ordinary least squares estimate,  $\hat{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ , by completing the square,

$$\begin{aligned}
 &(\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \\
 &= \mathbf{y}^T \mathbf{y} + \theta^T \mathbf{X}^T \mathbf{X} \theta - \theta^T \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \theta \\
 &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} + \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} + \theta^T \mathbf{X}^T \mathbf{X} \theta - \theta^T \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \theta \\
 &\quad + \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\
 &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} + \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} + \theta^T \mathbf{X}^T \mathbf{X} \theta \\
 &\quad - \theta^T \mathbf{X}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X} \theta + \mathbf{y}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\
 &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \hat{\theta} - \hat{\theta}^T \mathbf{X}^T \mathbf{y} + \hat{\theta}^T \mathbf{X}^T \mathbf{X} \hat{\theta} + \theta^T \mathbf{X}^T \mathbf{X} \theta - \theta^T \mathbf{X}^T \mathbf{X} \hat{\theta} - \hat{\theta}^T \mathbf{X}^T \mathbf{X} \theta + \hat{\theta}^T \mathbf{X}^T \mathbf{X} \hat{\theta} \\
 &= (\mathbf{y} - \mathbf{X} \hat{\theta})^T (\mathbf{y} - \mathbf{X} \hat{\theta}) + (\theta - \hat{\theta})^T \mathbf{X}^T \mathbf{X} (\theta - \hat{\theta}). \tag{5.7}
 \end{aligned}$$

Hence, it follows that the posterior distribution can be given as

$$\begin{aligned}
 p(\theta, \tau | \mathbf{y}, \mathbf{X}) &\propto \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \right\} \exp \left\{ -\frac{\tau}{2g} \theta^T \mathbf{X}^T \mathbf{X} \theta \right\} \\
 &= \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X} \hat{\theta})^T (\mathbf{y} - \mathbf{X} \hat{\theta}) - \frac{\tau}{2} (\theta - \hat{\theta})^T \mathbf{X}^T \mathbf{X} (\theta - \hat{\theta}) \right\} \exp \left\{ -\frac{\tau}{2g} \theta^T \mathbf{X}^T \mathbf{X} \theta \right\}. \tag{5.8}
 \end{aligned}$$

We can use this joint posterior distribution to determine the conditional posterior distribution of the parameters,  $p(\theta | \mathbf{y}, \tau)$ , the posterior distribution for the precision,  $p(\tau | \mathbf{y}, \mathbf{X})$ , as well as the unconditional distribution of the model parameters  $p(\theta | \mathbf{y}, \mathbf{X})$ . Each of these will now be derived in turn.

Firstly, the conditional posterior distribution of the parameters is given by

$$\begin{aligned}
 p(\theta | \mathbf{y}, \mathbf{X}, \tau) &= p(\theta, \tau | \mathbf{y}, \mathbf{X}) p(\tau) \\
 &\propto \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) \right\} \exp \left\{ -\frac{\tau}{2g} \theta^T \mathbf{X}^T \mathbf{X} \theta \right\}
 \end{aligned}$$

$$\begin{aligned}
&= \exp \left\{ -\frac{\tau}{2} (\mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta}) - \frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} \\
&= \exp \left\{ -\frac{\tau}{2} \left( \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \frac{g+1}{g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right) \right\} \\
&= \exp \left\{ -\frac{\tau}{2} \cdot \frac{(g+1)}{g} (\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y}) \right\}
\end{aligned} \tag{5.9}$$

which is the kernel for a Gaussian density with mean

$$\boldsymbol{\mu}_g = \frac{g+1}{g} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \frac{g+1}{g} \hat{\boldsymbol{\theta}} \tag{5.10}$$

and covariance matrix

$$\boldsymbol{\Sigma}_g = \frac{g}{g+1} (\tau \mathbf{X}^T \mathbf{X})^{-1} \tag{5.11}$$

following from Result 4.11.

Hence, the conditional posterior distribution for the model parameters using Zellner's  $g$ -prior is given by

$$p(\boldsymbol{\theta} | \mathbf{y}, \mathbf{X}, \tau) \sim \mathcal{N} \left( \frac{g}{g+1} \hat{\boldsymbol{\theta}}, \frac{g}{g+1} (\tau \mathbf{X}^T \mathbf{X})^{-1} \right). \tag{5.12}$$

Similarly, we can derive the posterior distribution for  $\tau$  as follows:

$$\begin{aligned}
p(\tau | \mathbf{y}, \mathbf{X}) &= \int p(\boldsymbol{\theta}, \tau | \mathbf{y}, \mathbf{X}) d\boldsymbol{\theta} \\
&= \int \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X} \hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X} \hat{\boldsymbol{\theta}}) - \frac{\tau}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{X}^T \mathbf{X} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\} \exp \left\{ -\frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} d\boldsymbol{\theta} \\
&= \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X} \hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X} \hat{\boldsymbol{\theta}}) \right\} \\
&\quad \cdot \int \exp \left\{ -\frac{\tau}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{X}^T \mathbf{X} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\} \exp \left\{ -\frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} d\boldsymbol{\theta}.
\end{aligned} \tag{5.13}$$

Now

$$\begin{aligned}
&\int \exp \left\{ -\frac{\tau}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{X}^T \mathbf{X} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\} \exp \left\{ -\frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} d\boldsymbol{\theta} \\
&= \int \exp \left\{ -\frac{\tau}{2} (\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}}) - \frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} d\boldsymbol{\theta} \\
&= \int \exp \left\{ -\frac{\tau}{2} \left( \frac{g+1}{g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right) \right\} d\boldsymbol{\theta} \\
&= \int \exp \left\{ -\frac{\tau(g+1)}{2g} \left( \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \frac{g}{g+1} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} - \frac{g}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \frac{g}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right) \right\} d\boldsymbol{\theta} \\
&= \int \exp \left\{ -\frac{\tau(g+1)}{2g} \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \mathbf{X}^T \mathbf{X} \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right) - \frac{\tau}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right\} d\boldsymbol{\theta}
\end{aligned}$$

$$\begin{aligned}
&= \exp \left\{ -\frac{\tau}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right\} \int \exp \left\{ -\frac{\tau(g+1)}{2g} \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \mathbf{X}^T \mathbf{X} \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right) \right\} d\boldsymbol{\theta} \\
&\propto \exp \left\{ -\frac{\tau}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right\} \left| \left( \frac{\tau(g+1)}{g} \mathbf{X}^T \mathbf{X} \right)^{-1} \right|^{\frac{1}{2}} \\
&\propto \exp \left\{ -\frac{\tau}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right\} \tau^{-\frac{p}{2}}
\end{aligned} \tag{5.14}$$

since we are only interested in terms relating to  $\tau$ . Hence,

$$\begin{aligned}
p(\tau|\mathbf{y}, \mathbf{X}) &= \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) \right\} \exp \left\{ -\frac{\tau}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right\} \tau^{-\frac{p}{2}} \\
&= \tau^{\frac{N}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) - \frac{\tau}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right\} \\
&= \tau^{\frac{N}{2}-1} \exp \left\{ -\tau \left[ \frac{1}{2} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) + \frac{1}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right] \right\}
\end{aligned} \tag{5.15}$$

which we recognize as an unnormalized Gamma distribution.

Thus,

$$p(\tau|\mathbf{y}, \mathbf{X}) \sim \text{Gamma} \left( \frac{N}{2}, \frac{s^2}{2} + \frac{1}{2(g+1)} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right) \tag{5.16}$$

where

$$s^2 = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}). \tag{5.17}$$

Similarly, the unconditional posterior distribution for the model parameters  $\boldsymbol{\theta}$  can be derived in an analogous manner to obtain a multivariate Student's  $t$ -distribution. Hence,

$$\begin{aligned}
p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) &= \int p(\boldsymbol{\theta}, \tau|\mathbf{y}, \mathbf{X}) d\tau \\
&= \int \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) - \frac{\tau}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{X}^T \mathbf{X} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\} \exp \left\{ -\frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} d\tau \\
&= \int \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) - \frac{\tau}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{X}^T \mathbf{X} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) - \frac{\tau}{2g} \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \right\} d\tau \\
&= \int \tau^{\frac{N+p}{2}-1} \exp \left\{ -\frac{\tau}{2} \left[ \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \left( \frac{g+1}{g} \mathbf{X}^T \mathbf{X} \right) \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right) + s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right] \right\} d\tau \\
&= \int \tau^{\frac{N+p}{2}-1} \exp \left\{ -\tau \frac{c}{2} \right\} d\tau \\
&= \frac{\Gamma \left( \frac{N+p}{2} \right)}{\left( \frac{c}{2} \right)^{\frac{N+p}{2}}}
\end{aligned}$$

$$\propto c^{-\frac{(N+p)}{2}} \quad (5.18)$$

since we recognize the integral in the third-last step above as the kernel of a Gamma distribution.

Now,

$$\begin{aligned} c^{-\frac{N+p}{2}} &= \left[ \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \left( \frac{g+1}{g} \mathbf{X}^T \mathbf{X} \right) \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right) + s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right]^{-\frac{(N+p)}{2}} \\ &= \left[ 1 + \frac{\left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \left( \frac{g+1}{g} \mathbf{X}^T \mathbf{X} \right) \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)}{s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}}} \right]^{-\frac{(N+p)}{2}} \\ &= \left[ 1 + \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \left( \left( s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right)^{-1} \frac{g+1}{g} \mathbf{X}^T \mathbf{X} \right) \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right) \right]^{-\frac{(N+p)}{2}} \\ &= \left[ 1 + \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \left( \left( s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right) \frac{g}{g+1} (\mathbf{X}^T \mathbf{X})^{-1} \right) \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right) \right]^{-\frac{(N+p)}{2}} \\ &= \left[ 1 + \frac{\left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)^T \left( \frac{g \left( s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right)}{N(g+1)} (\mathbf{X}^T \mathbf{X})^{-1} \right) \left( \boldsymbol{\theta} - \frac{g}{g+1} \hat{\boldsymbol{\theta}} \right)}{N} \right]^{-\frac{(N+p)}{2}} \end{aligned} \quad (5.19)$$

which is the unnormalized density of the multivariate Student's  $t$ -distribution. Hence,

$$\boldsymbol{\theta} | \mathbf{y}, \mathbf{X} \sim \mathcal{T} \left( N, \frac{g}{g+1} \hat{\boldsymbol{\theta}}, \frac{g \left( s^2 + \frac{1}{g+1} \hat{\boldsymbol{\theta}}^T \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\theta}} \right)}{N(g+1)} (\mathbf{X}^T \mathbf{X})^{-1} \right). \quad (5.20)$$

Since the aim with variable selection is to determine a subset of the original predictors and possibly improve the resulting fit of the model, we need to be able to compare the fits of each of these different models. In Section 4.3.1, it was discussed how the Bayes factor can be used for exactly this. To determine the Bayes factor for two competing models, we require the marginal likelihoods of each. In this variable selection context, these different models correspond to the different combinations in which each variable can be included or excluded from the model. The attractiveness of the  $g$ -prior is that all marginal likelihoods can be determined in closed form.

Since we know that

$$p(\boldsymbol{\theta} | \tau, \mathbf{X}) \sim \mathcal{N} \left( \mathbf{0}, \frac{g}{\tau} (\mathbf{X}^T \mathbf{X})^{-1} \right) \quad (5.21)$$



it follows immediately that (Zabaras, 2017)

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \tau^{-1}(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)). \quad (5.22)$$

Hence, the marginal likelihood can be obtained by simply integrating out the precision parameter  $\tau$ . Thus,

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, g) &= \int p(\mathbf{y}|\mathbf{X}, \tau, g)p(\tau) d\tau \\ &= \int (2\pi)^{-\frac{p}{2}} |\tau^{-1}(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)|^{-\frac{1}{2}} \exp\left\{-\frac{\tau}{2}\mathbf{y}^T(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^{-1}\mathbf{y}\right\} \{\tau^{-1}\} d\tau \\ &\propto |\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T|^{-\frac{1}{2}} \int \tau^{\frac{N}{2}-1} \exp\left\{-\frac{\tau}{2}\mathbf{y}^T(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^{-1}\mathbf{y}\right\} d\tau \\ &= |\mathbf{I}_p + g(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}|^{-\frac{1}{2}} \int \tau^{\frac{N}{2}-1} \exp\left\{-\frac{\tau}{2}\mathbf{y}^T(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^{-1}\mathbf{y}\right\} d\tau \\ &= |\mathbf{I}_p + g\mathbf{I}_p|^{-\frac{1}{2}} \int \tau^{\frac{N}{2}-1} \exp\left\{-\frac{\tau}{2}\mathbf{y}^T(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^{-1}\mathbf{y}\right\} d\tau \\ &= |(1+g)\mathbf{I}_p|^{-\frac{1}{2}} \int \tau^{\frac{N}{2}-1} \exp\left\{-\frac{\tau}{2}\mathbf{y}^T(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^{-1}\mathbf{y}\right\} d\tau \\ &\propto (1+g)^{-\frac{p}{2}} [\mathbf{y}^T(\mathbf{I}_N + g\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^{-1}\mathbf{y}]^{-\frac{N}{2}} \\ &= (1+g)^{-\frac{p}{2}} \left[\mathbf{y}^T \left(\mathbf{I}_N - \frac{g}{g+1}\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\right) \mathbf{y}\right]^{-\frac{N}{2}} \\ &= (1+g)^{-\frac{p}{2}} \left[\mathbf{y}^T \mathbf{y} - \frac{g}{g+1}\mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\right]^{-\frac{N}{2}}. \end{aligned} \quad (5.23)$$

Hence, we have obtained a closed form solution for the marginal likelihood which means that the Bayes factor for any two competing models can easily be calculated (Nielsen *et al.*, 2014). This above marginal likelihood in Equation 5.23 is often expressed in the literature in the following equivalent form:

$$p(\mathbf{y}|\mathbf{X}, g) \propto (\mathbf{y}^T\mathbf{y})^{-\frac{N-1}{2}} \frac{(1+g)^{\frac{N-p-1}{2}}}{(1+g(1-R^2))^{\frac{N-1}{2}}} \quad (5.24)$$

where

$$R^2 = 1 - \frac{(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}})}{\mathbf{y}^T\mathbf{y}}. \quad (5.25)$$

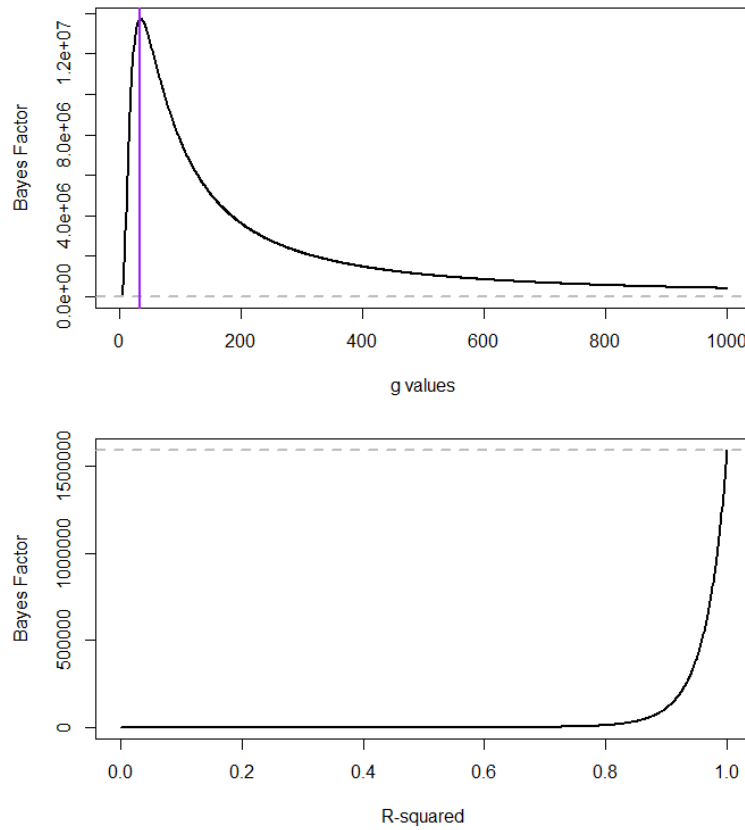
Thus, the Bayes factor for comparing any model, say  $\mathcal{M}_k$ , to the null model,  $\mathcal{M}_0$ , is given by

$$\begin{aligned}
B_{k0} &= \frac{\left[ (\mathbf{y}^T \mathbf{y})^{-\frac{N-1}{2}} \frac{(1+g)^{\frac{N-p_k-1}{2}}}{\left(1+g(1-R_k^2)\right)^{\frac{N-1}{2}}} \right]}{\left[ (\mathbf{y}^T \mathbf{y})^{-\frac{N-1}{2}} \frac{(1+g)^{\frac{N-1}{2}}}{\left(1+g(1-0)\right)^{\frac{N-1}{2}}} \right]} \\
&= (1+g)^{\frac{N-p_k-1}{2}} \left(1+g(1-R_k^2)\right)^{-\frac{N-1}{2}} \quad (5.26)
\end{aligned}$$

where  $p_k$  is the number of predictors in the model  $\mathcal{M}_k$  and  $R_k$  is defined as in Equation 5.25 except using  $\mathbf{X}_k$  and  $\boldsymbol{\theta}_k$  corresponding to the  $k$  predictors included in model  $\mathcal{M}_k$  and their associated parameter coefficients.

The natural question which now arises is what value should be chosen for the constant  $g$ . We notice some interesting properties relating to certain values of  $g$ . Firstly, if  $g \rightarrow 0$ , the posterior distribution tends towards the prior distribution and so nothing is achieved. On the other hand, if  $g \rightarrow \infty$ , we obtain a flat, uniform prior and so the posterior corresponds to the ordinary least squares result. However, in this case, a large value of  $g$  will result in the Bayes factor obtained from comparing any model to the null model tending towards zero. Hence, the null model will always be favoured. This is referred to as Lindley's paradox (Lindley, 1957). Alternatively, if there is a specific model, say,  $\mathcal{M}_k$ , for which the data shows overwhelming favour such that  $R_k^2 \rightarrow 1$ , if all other values remain constant, the value of the Bayes factor will tend towards the constant  $(1+g)^{(N-p_k-1)/2}$ . This phenomenon is referred to as the information paradox since we would have expected the Bayes factor to rather tend towards infinity if the data showed such strong evidence towards that model (Liang *et al.*, 2008).

Hence, the choice of  $g$  clearly has a large influence on the chosen model by controlling the amount by which the prior distribution and the likelihood of the data contribute to the resulting posterior. These two paradoxes are displayed visually in Figure 5.1. For demonstration purposes, fixed, arbitrary values for the marginal likelihood function were chosen. In particular, we set  $N = 30$ ,  $p_k = 3$  and  $R_k^2 = 0.8$  and varied  $g$  from 0 to 1000 to demonstrate Lindley's paradox. To demonstrate the information paradox, we fixed  $g = 2$  and varied  $R^2$  from 0 to 1 using the same values for the other constants as before. The top plot in Figure 5.1 demonstrates Lindley's paradox and the bottom plot demonstrates the information paradox. The vertical purple line in the top plot is the local empirical Bayes estimate for  $g$  which will be discussed shortly.



**Figure 5.1: Simulated example of Lindley's paradox and the information paradox.**

Originally, Zellner (Zellner, cited in Liang *et al.*, 2008) suggested the choice  $g = N$ , which is referred to as the unit information prior (UIP), since the amount of information in the prior is then equal to the amount of information contained in one data observation. This follows since the covariance for the  $g$ -prior is then exactly equal to the inverse of the expected Fisher information divided by  $N$ , and it was discussed that the Fisher information reflects the amount of information that the entire dataset provides about the unknown parameter set. Since then, several other proposed values of  $g$  have been studied to try and avoid some of the undesirable paradoxes that can occur with certain choices, such as those mentioned above.

One natural proposal for a Bayesian variable selection problem was to use a fully Bayesian approach and treat the hyperparameter  $g$  as a random variable. The first use of this was by Zellner and Siow (1980) termed the Zellner-Siow prior, where the prior for  $g$  was modelled using an inverse gamma distribution, that is,

$$g \sim \text{InvGamma}\left(\frac{1}{2}, \frac{N}{2}\right). \quad (5.27)$$

This avoided the aforementioned issues that arose with a fixed value for  $g$  but did not initially become as widely adopted since a closed form solution for the marginal likelihood no longer existed and it was exactly this possibility of a closed form solution for the marginal likelihood that prompted the favoured use of Zellner's  $g$ -prior. George & Foster (2000) further proposed an

empirical Bayes (EB) approach for determining  $g$  which meant that the value of  $g$  became dependent on the dataset of interest. Using this method, either a local or global empirical Bayes approach was suggested. The local EB approach allows the value of  $g$  to vary for each model by selecting the value of  $g$  which corresponds to the maximum likelihood estimate of the marginal likelihood function (constrained to be non-negative). That is, we assign  $g_k$  to each model where

$$g_k^{LEB} = \underset{g>0}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{X}, g, \mathcal{M}_k)$$

$$= \underset{g>0}{\operatorname{argmax}} \left\{ (\mathbf{y}^T \mathbf{y})^{-\frac{N-1}{2}} \frac{(1+g)^{\frac{N-p_k-1}{2}}}{\left(1+g(1-R_k^2)\right)^{\frac{N-1}{2}}} \right\}. \quad (5.28)$$

To determine the value of  $g$  which maximizes this quantity, we can equivalently determine the value of  $g$  which maximizes the natural logarithm of this quantity. Hence,

$$\begin{aligned} 0 &= \frac{\partial}{\partial g} \log \left[ \frac{(1+g)^{\frac{N-p_k-1}{2}}}{\left(1+g(1-R_k^2)\right)^{\frac{N-1}{2}}} \right] \\ \Rightarrow 0 &= \frac{N-p_k-1}{2(1+g^{LEB})} - \frac{(N-1)(1-R_k^2)}{2(1+g^{LEB}(1-R_k^2))} \\ \Rightarrow \frac{N-p_k-1}{2(1+g^{LEB})} &= \frac{(N-1)(1-R_k^2)}{2(1+g^{LEB}(1-R_k^2))} \\ \Rightarrow (1+g^{LEB}(1-R_k^2))((N-1)-p_k) &= (1+g^{LEB})(N-1)(1-R_k^2) \\ \Rightarrow g^{LEB}[(N-1)(1-R_k^2)-p_k(1-R_k^2)-(N-1)(1-R_k^2)] &= -(N-1)+p_k+(N-1)(1-R_k^2) \\ \Rightarrow g^{LEB} &= \frac{-(N-1)+p_k+(N-1)(1-R_k^2)}{(N-1)(1-R_k^2)-p_k(1-R_k^2)-(N-1)(1-R_k^2)} \\ \Rightarrow g^{LEB} &= \frac{\left[-\frac{1}{(1-R_k^2)} + \frac{p_k}{(N-1)(1-R_k^2)} + 1\right]}{\left[1 - \frac{p_k}{N-1} - 1\right]} \\ \Rightarrow g^{LEB} &= \frac{(N-1)}{p_k(1-R_k^2)} - \frac{1}{(1-R_k^2)} - \frac{N-1}{p_k} \\ \Rightarrow g^{LEB} &= \frac{p_k + NR_k^2 - R_k^2}{p_k(1-R_k^2)} \\ \Rightarrow g^{LEB} &= \frac{R_k^2 N - R_k^2 - p_k R_k^2 + p_k + p_k R_k^2}{p_k(1-R_k^2)} \\ \Rightarrow g^{LEB} &= \frac{\left[\frac{R_k^2(N-1-p_k)-p_k(1-R_k^2)}{p_k(N-1-p_k)}\right]}{\left[\frac{1-R_k^2}{N-1-p_k}\right]} \\ \Rightarrow g^{LEB} &= \frac{\left[\frac{R_k^2}{p_k} - \frac{1-R_k^2}{N-1-p_k}\right]}{\left[\frac{1-R_k^2}{N-1-p_k}\right]} \end{aligned}$$

$$\begin{aligned} \Rightarrow g^{LEB} &= \frac{\left[ \frac{R_k^2}{p_k} \right]}{\left[ \frac{1 - R_k^2}{N - 1 - p_k} \right]} - 1 \\ \Rightarrow g^{LEB} &= F_k - 1 \end{aligned} \quad (5.29)$$

where  $F_k$  is the normal  $F$ -statistic used in the frequentist approach of testing the hypothesis  $\theta_k = 0$ , given by

$$F_k = \frac{\left[ R_k^2 / p_k \right]}{\left[ (1 - R_k^2) / (N - p_k - 1) \right]}. \quad (5.30)$$

Hence, the local empirical Bayes method sets each  $g_k$  to

$$g_k^{LEB} = \max\{0, F_k - 1\}. \quad (5.31)$$

Referring back to Figure 5.1, the vertical purple line represents this local empirical bayes estimate for  $g$  which, in this simple example, is indeed the value of  $g$  which maximizes the Bayes factor.

On the other hand, the global EB approach selects a value of  $g$  that is constant over all models. Here,  $g$  is found by determining the maximum of the weighted sum of all marginal likelihoods where the weights are given by the prior probabilities for each of the models. Thus, the global EB estimate of  $g$  is given by

$$g^{GEB} = \operatorname{argmax}_{g>0} p(\mathcal{M}_k) p(\mathbf{y}|\mathbf{X}, g, \mathcal{M}_k). \quad (5.32)$$

Unlike the local EB approach, this solution does not exist in closed form and the EM algorithm is typically implemented to approximate this value. Of course, a drawback of an EB approach in general compared to any fully Bayesian approach is that the uncertainty in the estimates are not incorporated into the estimation since a point estimate is returned.

### 5.3 THE SPIKE-AND-SLAB PRIOR

Another example of a prior distribution that leads to variable selection is the spike-and-slab prior proposed by Mitchell & Beauchamp (1988). By using this prior and (possibly) eliminating some variables from the model, we obtain a sparser and more parsimonious model that is easier to work with and interpret without having compromised on predictive performance. The idea behind the spike-and-slab prior is that the spike component has a concentrated density for values close to zero – allowing for shrinkage towards zero to occur – and the wide slab represents all other possible (non-zero) values that the regression coefficients could take (Malsiner-Walli & Wagner, 2016).

In Section 4.2, we worked with an example where the likelihood function for the data was given by

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \tau^{-1}\mathbf{I}) \quad (5.33)$$

and we now further introduce a set of auxiliary indicator variables,  $s_k \in \{0,1\}, k = 1, \dots, p$  to define the spike-and-slab prior. These auxiliary variables induce sparsity into the model by controlling whether certain parameters are included in the model or not. That is, when  $s_k = 0$ , then the corresponding parameter  $\theta_k$ , and thus variable  $x_k$ , is removed from the model by allocating it to the spike component of the prior. Similarly, if  $s_k = 1$ , the corresponding parameter  $\theta_k$  is allocated to the slab component of the prior resulting in a non-zero estimate for the parameter. The joint distribution of the indicator variables is given by a Bernoulli distribution,

$$p(\mathbf{s}) = \prod_{k=1}^p \eta^{s_k} (1 - \eta)^{1-s_k} \quad (5.34)$$

where  $0 \leq \eta \leq 1$  specifies a prior level of sparsity such that  $P(s_k = 1) = \eta$  (Theodoridis, 2015:660).

Smaller values of  $\eta$  will thus correspond to a higher degree of sparsity in the model due to this Bernoulli distribution. Typically, we set  $\eta = \frac{1}{2}$  which is commonly referred to as an indifference, or uniform prior, since we are suggesting no prior preference to whether a variable should be included in the model or not. Alternatively, any other distribution for these probabilities could also be used to represent prior beliefs about inclusion of the individual variables.

There are two different types of spikes that are used in practice: either the spike is given by a point mass at zero, which is referred to as a Dirac spike, or the spike component is any unimodal distribution with a concentrated mode close to zero (Malsiner-Walli & Wagner, 2016). In both cases, the spike-and-slab prior consists of a mixture of distributions on the regression coefficients and takes the form

$$p(\boldsymbol{\theta}) = \prod_{k=1}^p \left( s_k p_{\text{slab}}(\theta_k) + (1 - s_k) p_{\text{spike}}(\theta_k) \right). \quad (5.35)$$

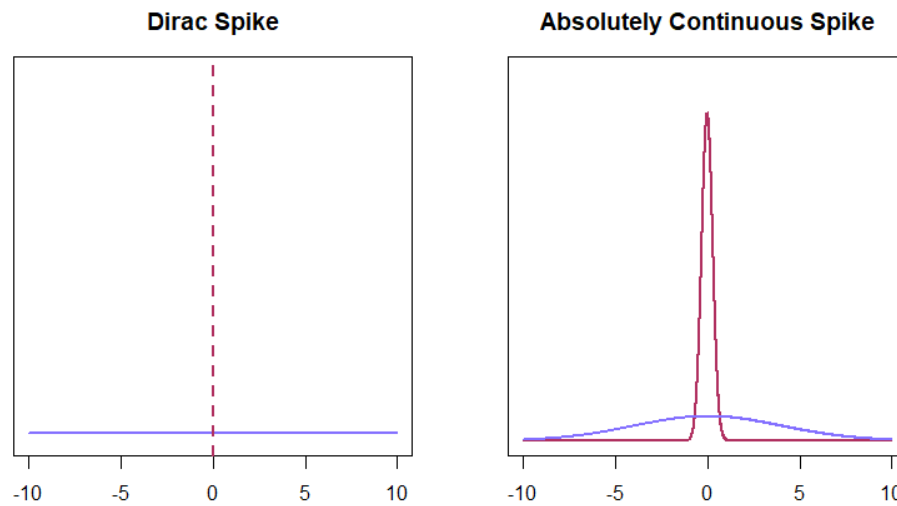
The Dirac spike is what was initially proposed by Mitchell & Beauchamp (1988) where the prior distribution for the spike is given by  $p_{\text{spike}}(\theta_k) = \delta(\theta_k)$  with  $\delta(\cdot)$  denoting the Dirac delta function. With this formulation, the coefficients can be shrunk down to exactly zero and subsequently removed from the model. The slab component was defined as a uniform distribution between two extreme limits.

This idea was then adapted slightly by George & McCulloch (1993) where they proposed an absolutely continuous spike. In this case, both the spike and slab components are assumed to be Gaussian densities and so the prior distribution function is a mixture of two Gaussian density functions, and takes the form

$$p(\boldsymbol{\theta}) = \prod_{k=1}^p \left( s_k \mathcal{N}(0, c_k^2 r_k^2) + (1 - s_k) \mathcal{N}(0, r_k^2) \right) \quad (5.36)$$

where  $c_k > 1$  is large and  $r_k > 0$  is small, such that  $c_k^2 r_k^2$  is still large. The motivation for this is as follows: when  $s_k = 1$  the prior for  $\theta_k$  is given by  $\mathcal{N}(0, c_k^2 r_k^2)$  which corresponds to a very flat Gaussian distribution due to the large variance, representing all the possible (non-zero) values for the parameter coefficient; when  $s_k = 0$  and the variable should be removed from the model, the distribution of the coefficient will be given by  $\theta_k \sim \mathcal{N}(0, r_k^2)$  which is a Gaussian distribution sharply peaked around zero. It is important to note that, unlike when the Dirac spike is used, the coefficients will not be shrunk exactly to zero, only very small values. It is recommended that  $r_k$  and  $c_k$  be treated as tuning parameters which will vary based on the specific task on hand.

Figure 5.2 displays examples each of these formulations of the spike-and-slab prior. The left plot shows the Dirac spike, and the right plot shows the absolutely continuous spike.



**Figure 5.2: Dirac and absolutely continuous spikes for a spike-and-slab prior.**

Other variants of the spike-and-slab prior also exist where minor adjustments have been made to either the prior distribution for the spike, or the slab, or both. For example, Ishwaran & Rao (2005) also made use of a continuous bimodal distribution, but further proposed the use of a rescaled spike-and-slab prior. The motivation for this was to construct a spike-and-slab prior that is more robust to being dominated by the likelihood function in large sample settings since otherwise variable selection will most likely not occur due to the prior distribution having very little influence on the posterior. Kuo & Mallick (1998) proposed a similar method to that of George & McCulloch (1993), but instead of building a hierarchical model, they incorporated indicator variables into the linear regression equation which included all  $2^p$  possible models. This also allows for easy incorporation of any interaction effects into the model. In all these different variants of spike-and-

slab priors, the resulting posterior distribution does not exist in closed form since this prior is not conjugate. Hence, Gibbs sampling is typically used to obtain approximations to the posterior distributions (Theodoridis, 2015:657).

## 5.4 THE BAYESIAN LASSO & RIDGE REGRESSION

One of the most popular frequentist approaches to variable selection is subset selection. This includes techniques such as best subset selection or forward- and backward-stepwise selection. These methods are useful since they completely discard certain predictors from the model. However, these methods tend to be associated with a higher variability in the results and so prediction accuracy often suffers. This motivated the use of shrinkage methods, with the more popular shrinkage techniques being ridge regression and the LASSO.

Both ridge regression and the LASSO shrink the coefficient estimates towards zero by imposing a penalty on their respective sizes. These two methods are special cases of the following general penalization criterion:

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmin}} \left\{ (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) + \lambda \sum_{i=1}^p |\theta_i|^q \right\}, \quad q \geq 0. \quad (5.37)$$

Here,  $\lambda \geq 0$  is a tuning parameter that controls the amount of penalty imposed on the coefficients. Clearly, a value of  $\lambda = 0$  corresponds to no penalization and the ordinary least squares estimate will be obtained. Similarly, as  $\lambda \rightarrow \infty$ , the coefficients are forced to zero and we obtain the constant model equal to the mean of the responses (Hastie *et al.*, 2009:72). This formulation assumes without loss of generality that the data has been standardized so that each column has a zero mean and the intercept term is discarded. The LASSO and ridge regression are obtained from the special cases when  $q = 1$  and  $q = 2$  in Equation 5.37, respectively.

### 5.4.1 A Bayesian Perspective of Ridge Regression

As mentioned, the formulation of the ridge regression problem is given by Equation 5.37 with  $q = 2$ . That is, the coefficients which satisfy

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmin}} \left\{ (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) + \lambda \sum_{i=1}^p \theta_i^2 \right\} \quad (5.38)$$

correspond to the ridge regression estimates. The solution for this problem is easy to derive. Consider

$$(\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) + \lambda \sum_{i=1}^p \theta_i^2 = (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) + \lambda \theta^T \theta$$



$$\begin{aligned}
&= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta} \\
&= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X} \boldsymbol{\theta} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}.
\end{aligned} \tag{5.39}$$

Hence, to determine the minimum, we take the derivative of this expression with respect to  $\boldsymbol{\theta}$ , set equal to zero and solve. Therefore,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \{ \mathbf{y}^T \mathbf{y} - 2\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta} \} &= -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} + \lambda \mathbf{I} \\
&\Rightarrow -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\theta}} + 2\lambda \tilde{\boldsymbol{\theta}} = \mathbf{0} \\
&\Rightarrow \tilde{\boldsymbol{\theta}} (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) = \mathbf{X}^T \mathbf{y} \\
&\Rightarrow \tilde{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1}.
\end{aligned} \tag{5.40}$$

Hastie *et al.* (2009: 65) recognized that this ridge regression solution could similarly have been derived as the MAP estimator from a Bayesian approach to linear regression. Formally, if we assign the zero mean Gaussian prior

$$p(\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \tag{5.41}$$

to a dataset with likelihood function

$$p(\mathbf{y}|\boldsymbol{\theta}, \tau) \sim \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \tau^{-1} \mathbf{I}) \tag{5.42}$$

with known values of  $\sigma^2$  and  $\tau$ , then, the posterior distribution of these parameters (which was derived in Chapter 4 using Bayes' rule) is given by

$$p(\boldsymbol{\theta}|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N). \tag{5.43}$$

The posterior covariance matrix and mean vector are given respectively by

$$\boldsymbol{\Sigma}_N^{-1} = \tau \mathbf{X}^T \mathbf{X} + \frac{1}{\sigma^2} \mathbf{I} \quad \text{and} \quad \boldsymbol{\mu}_N = \tau \boldsymbol{\Sigma}_N \mathbf{X}^T \mathbf{y}. \tag{5.44}$$

The MAP estimate for this distribution is equivalent to the posterior mean (or median since this is a Gaussian distribution). Considering this posterior mean, we can expand the expression and rearrange the terms to obtain:

$$\begin{aligned}
\boldsymbol{\mu}_N &= \tau \boldsymbol{\Sigma}_N \mathbf{X}^T \mathbf{y} \\
&= \tau \left( \tau \mathbf{X}^T \mathbf{X} + \frac{1}{\sigma^2} \mathbf{I} \right)^{-1} \mathbf{X}^T \mathbf{y} \\
&= \left( \mathbf{X}^T \mathbf{X} + \frac{1}{\tau \sigma^2} \mathbf{I} \right)^{-1} \mathbf{X}^T \mathbf{y}
\end{aligned} \tag{5.45}$$

which is exactly the form of the ridge regression solution with  $\lambda = \frac{1}{\tau \sigma^2}$ .

Hence, this confirms that the ridge regression problem can equivalently be regarded as a Bayesian approach to linear regression whereby the regression coefficients are obtained as the

MAP estimates when a zero-mean, white noise Gaussian prior is used. Furthermore, this implies that the MAP estimator is simply a regularized version of the least squares estimate where this regularization is imposed via the prior mean and variance (Theodoridis, 2015:588).

### 5.4.2 The Bayesian LASSO

The ridge regression problem was the one obtained when  $q$  was set equal to 2 in Equation 5.37. One of the drawbacks of the ridge regression parameter estimates is that none of them will ever be shrunk exactly to zero. Those variables which can be excluded from the model will simply be shrunk to very small values close to zero. This is what motivated the use of the LASSO in favour of ridge regression. The LASSO problem is formulated by now using  $q = 1$ , that is, the coefficient estimates are determined by

$$\tilde{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda \sum_{i=1}^p |\theta_i| \right\}. \quad (5.46)$$

This formulation ensures that the regression coefficients will be shrunk exactly to zero and removed from the model entirely. Although no closed form solution exists for the LASSO, the least angle regression (LARS) algorithm (Efron *et al.*, 2004), with specific modifications for the LASSO, provides an efficient method for determining the entire solution path of the LASSO estimates at each step of the algorithm. This allows for visual representation of the changing parameter estimates as a function of the tuning parameter  $\lambda$ .

Park & Casella (2008) were the first to recognize that the form of the penalty term in the LASSO model suggested that LASSO estimates can similarly be interpreted as MAP estimates in the situation when the prior distribution placed on each parameter coefficient is an independent and identical Laplace distribution. The general form of a univariate Laplace (or double exponential) density function with parameters  $a$  and  $b$  is given by

$$p(\beta|a, b) = \frac{1}{2b} \exp \left\{ -\frac{|\beta - a|}{b} \right\}, \quad a \in \mathbb{R}, b > 0 \quad (5.47)$$

where the mean is given by  $a$  and the variance by  $2b^2$ . Thus, assigning the following Laplace prior to each parameter

$$\theta_i \sim \text{Laplace} \left( 0, \frac{1}{\tau\lambda} \right) \quad (5.48)$$

results in a posterior distribution whose mode corresponds to the LASSO solution. Hence, Park & Casella (2008) set the conditional prior distribution for  $\boldsymbol{\theta}$  to be

$$p(\boldsymbol{\theta}|\tau) = \prod_{i=1}^p \frac{\lambda\sqrt{\tau}}{2} \exp\{-\lambda\sqrt{\tau}|\theta_i|\}$$

$$= \left(\frac{\lambda\sqrt{\tau}}{2}\right)^p \exp\left\{-\lambda\sqrt{\tau} \sum_{i=1}^p |\theta_i|\right\} \quad (5.49)$$

where, for a fully Bayesian approach, Jeffrey's noninformative prior is placed on  $\tau$ , that is,

$$p(\tau) \propto \frac{1}{\tau}. \quad (5.50)$$

This conditional prior ensures that the resulting posterior distribution is unimodal since multimodality slows down the convergence of the Gibbs sampler and makes a MAP estimate less meaningful. Hence, they proposed the following hierarchical model:

$$\begin{aligned} \mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \tau &\sim \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \tau^{-1}\mathbf{I}_N) \\ \boldsymbol{\theta}|\tau, r_1^2, \dots, r_p^2 &\sim \mathcal{N}(\mathbf{0}, \tau^{-1}\mathbf{D}_r), \quad \text{where } \mathbf{D}_r = \text{diag}(r_1^2, \dots, r_p^2) \\ \tau, r_1^2, \dots, r_p^2 &\sim p(\tau) \prod_{i=1}^p \frac{\lambda^2}{2} e^{-\frac{\lambda^2 r_i^2}{2}} \\ \tau, r_1^2, \dots, r_p^2 &> 0. \end{aligned} \quad (5.51)$$

This hierarchical model results in the same conditional prior distribution of Equation 5.49 since Andrews and Mallows (1974) showed that the Laplace distribution can be expressed as a scaled mixture of normal densities, that is,

$$\frac{a}{2} e^{-a|z|} = \int \left\{ \frac{1}{\sqrt{2\pi s}} e^{-\frac{z^2}{2s}} \right\} \left\{ \frac{a^2}{2} e^{-\frac{a^2 s}{2}} \right\} ds, \quad a > 0. \quad (5.52)$$

To show this,

$$\begin{aligned} p(\boldsymbol{\theta}|\tau) &= \int p(\boldsymbol{\theta}|\tau, r_1^2, \dots, r_p^2) p(\tau, r_1^2, \dots, r_p^2) dr_1^2, \dots, r_p^2 \\ &= \int (2\pi)^{-\frac{p}{2}} |\tau^{-1}\mathbf{D}_r|^{-\frac{1}{2}} \exp\left\{-\frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{D}_r^{-1} \boldsymbol{\theta}\right\} \prod_{i=1}^p \frac{\lambda^2}{2} e^{-\frac{\lambda^2 r_i^2}{2}} dr_1^2, \dots, r_p^2 \\ &= \int (2\pi)^{-\frac{p}{2}} \tau^{\frac{p}{2}} |\mathbf{D}_r|^{-\frac{1}{2}} \exp\left\{-\frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{D}_r^{-1} \boldsymbol{\theta}\right\} \prod_{i=1}^p \frac{\lambda^2}{2} e^{-\frac{\lambda^2 r_i^2}{2}} dr_1^2, \dots, r_p^2 \\ &= \tau^{\frac{p}{2}} \int \prod_{i=1}^p \frac{1}{r_i \sqrt{2\pi}} \exp\left\{-\frac{\tau}{2} \boldsymbol{\theta}^T \mathbf{D}_r^{-1} \boldsymbol{\theta}\right\} \prod_{i=1}^p \frac{\lambda^2}{2} e^{-\frac{\lambda^2 r_i^2}{2}} dr_1^2, \dots, r_p^2 \\ &= \tau^{\frac{p}{2}} \prod_{i=1}^p \int \frac{1}{r_i \sqrt{2\pi}} \exp\left\{-\frac{\tau \theta_i^2}{2 r_i^2}\right\} \frac{\lambda^2}{2} \exp\left\{-\frac{\lambda^2 r_i^2}{2}\right\} dr_i^2 \\ &= \tau^{\frac{p}{2}} \prod_{i=1}^p \frac{\lambda}{2} \exp\{-\lambda\sqrt{\tau}|\theta_i|\}, \quad \text{from (5.52)} \end{aligned}$$

$$= \left( \frac{\lambda \sqrt{\tau}}{2} \right)^p \exp \left\{ -\lambda \sqrt{\tau} \sum_{i=1}^p |\theta_i| \right\} \quad (5.53)$$

as desired.

Since the prior distribution for the parameters contains the constant  $\lambda$ , methods are required to determine the optimal value for this hyperparameter. There are two proposed methods for doing this – either an empirical Bayes approach can be followed or  $\lambda$  can be treated as a random variable with a hyperprior assigned to it.

The EB approach in this case is in fact empirical Bayes Gibbs sampling introduced by Casella (2001). This method is essentially an adaption of the EM algorithm, whereby the model parameters are treated as the latent data and the hyperparameters are updated with each iteration of the algorithm. The Gibbs sampling component comes in as a means of estimating the expected values within the expectation step of the algorithm.

Alternatively, to avoid selecting a specific value for  $\lambda$  and instead using a fully Bayesian approach, a prior distribution for this parameter can be assigned. The Gamma distribution is a typical choice since it is conjugate to the likelihood for the LASSO model. The class of Gamma density functions takes the form,

$$p(\lambda|r, \delta) = \frac{\delta}{\Gamma(a)} \lambda^{a-1} \exp(-\delta\lambda), \quad \lambda, a, \delta > 0. \quad (5.54)$$

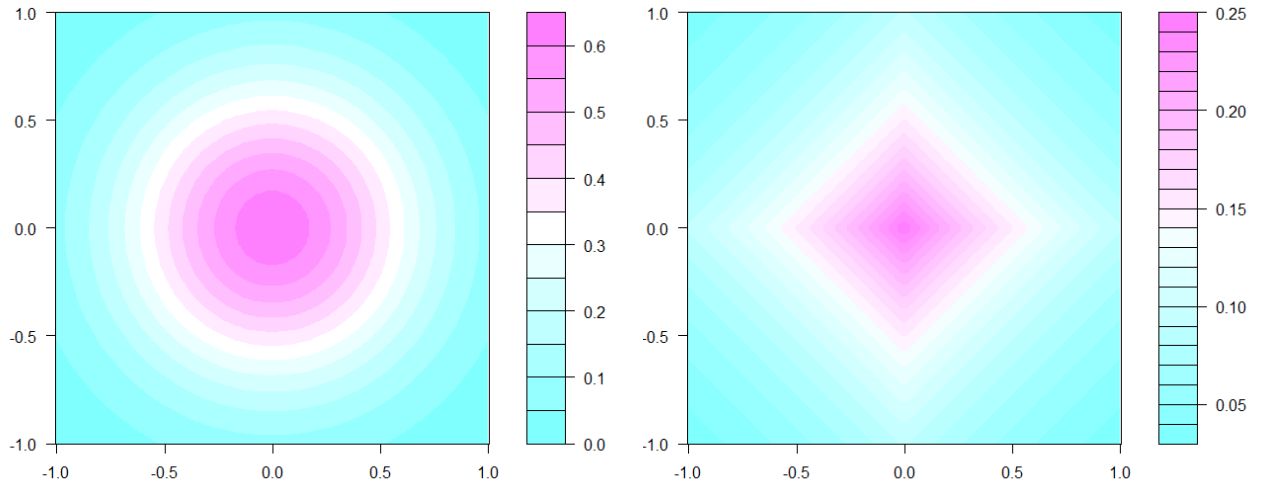
Further, the choice of prior should be such that the density approaches zero as  $\lambda \rightarrow \infty$  and also be essentially flat in all areas except those corresponding to the maximum likelihood estimates. For mathematical convenience,  $\lambda^2$  is considered in place of  $\lambda$ . This is done to allow for easy marginalization over  $\lambda^2$  by using a transformation back to the original variable  $\lambda$  (Park & Casella, 2008). The posterior distribution for  $\lambda^2$  will hence be given by

$$\begin{aligned} p(\lambda^2|\boldsymbol{\theta}, \tau, a, \delta) &= p(\boldsymbol{\theta}|\lambda^2, \tau) p(\lambda^2|a, \delta) \\ &= \left( \frac{\lambda^2 \tau}{2} \right)^p \exp \left\{ -\lambda^2 \tau \sum_{i=1}^p |\theta_i| \right\} \frac{\delta}{\Gamma(a)} \lambda^{2(a-1)} \exp\{-\delta\lambda^2\} \\ &\propto \lambda^{2(p+a-1)} \exp \left\{ -\lambda^2 \tau \sum_{i=1}^p |\theta_i| - \delta\lambda^2 \right\} \\ &= \lambda^{2(p+a-1)} \exp \left\{ -\lambda^2 \left( \tau \sum_{i=1}^p |\theta_i| + \delta \right) \right\}. \end{aligned} \quad (5.55)$$

Thus, we see that the conditional posterior distribution for  $\lambda^2$  is again a Gamma distribution, specifically,

$$p(\lambda^2 | \theta, \tau, a, \delta) \sim \text{Gamma} \left( p + a, \tau \sum_{i=1}^p |\theta_i| + \delta \right). \quad (5.56)$$

Hence, we see that both ridge regression and the LASSO can be interpreted from a Bayesian perspective depending on which prior distribution is assigned; for Bayesian ridge regression we assign a zero mean Gaussian prior and for the Bayesian LASSO we assign a Laplace prior. The contours in Figure 5.3 depict these prior distributions in two-dimensions.



**Figure 5.3: Gaussian and Laplace prior distributions in two dimensions.**

The regions in Figure 5.3 constructed by the contours correspond to the constraints imposed by the ridge and LASSO problems. Thus, in two dimensions, the constrained region for ridge regression is given by a circle and the constrained region for the LASSO is given by a diamond. The solutions to the optimization problem given in 5.37 are given by elliptical contours (defining a surface) increasing outwards from the least squares solution (i.e. when  $\lambda = 0$ ). The respective solutions for each of these optimization problems will be given by the first point in which these elliptical contours intersect the constrained region (Hastie *et al.*, 2009: 72).

Hence, it is clear from Figure 5.3 why the classical LASSO approach shrinks the coefficient estimates directly to zero. This is due to the corners that are present in the constrained region – a solution occurring at a corner will result in one of the coefficient estimates being exactly zero. In higher dimensions, there will be a greater number of corners for the solutions to hit and hence a higher probability for the LASSO to shrink coefficients to zero. Since the constrained region for ridge regression is a disk shape, the coefficient estimates will never be shrunk exactly to zero since it is very unlikely that this first point of contact with the constrained region will be on an axis.

It is important to note, however, that the Bayesian LASSO does not have this property of complete shrinkage to zero and so it is not strictly a variable selection technique, merely another shrinkage method whereby those parameter estimates which have little influence in the model will simply

have a posterior distribution with a high concentration near zero. However, reversible jump MCMC can be augmented into the MCMC sampling for both Bayesian ridge regression and the Bayesian LASSO to perform variable selection explicitly.

## 5.5 APPLICATIONS

The four variable selection techniques that have been discussed will now be compared in a single example. The prostate dataset obtained from the ElemStatLearn R package (Hastie *et al.*, 2019) contains 97 observations on 8 predictor variables and the aim is to predict the logarithm of the level of prostate specific antigen (PSA) based on these 8 variables. The data was split randomly into a training set (70%) and a held-out test set (30%).

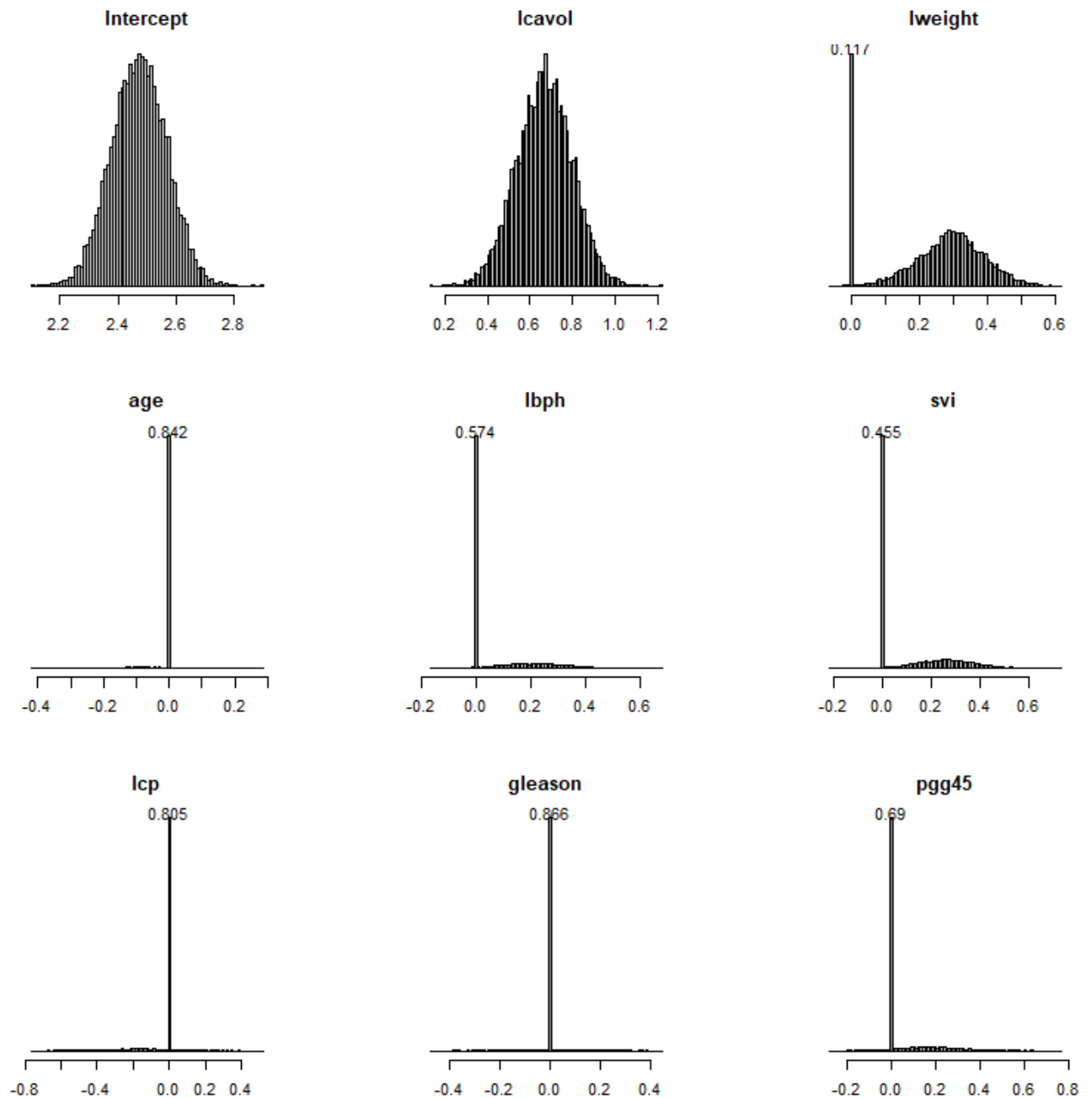
Firstly, two different  $g$ -priors were assigned to the dataset. The first was using a constant value of  $g = N$  as suggested in the original proposal by Zellner, and the second was using the Zellner-Siow inverse Gamma prior for  $g$ . Table 5.1 represents the inclusion probabilities for each of these two models. ‘HPM’ stands for highest posterior probability model and ‘MPM’ stands for median probability model. The stars indicate which of the variables have an inclusion probability of greater than 0.5 since these variables are then considered significant enough to be included in the model.

**Table 5.1: Inclusion probabilities for the two variants of Zellner’s  $g$ -prior.**

	Constant: $g = N$			Random: Zellner-Siow Prior		
	Inclusion Probability	HPM	MPM	Inclusion Probability	HPM	MPM
<b>lcavol</b>	1.000	*	*	1.000	*	*
<b>lweight</b>	0.8830	*	*	0.8852	*	*
<b>age</b>	0.1617			0.1961		
<b>lbph</b>	0.4271			0.4608		
<b>svi</b>	0.5435		*	0.5726		*
<b>lcp</b>	0.1889			0.2303		
<b>gleason</b>	0.1366			0.1623		
<b>pgg45</b>	0.3062			0.3422		

Using the MPM to select the best model, both approaches find only 3 variables to be significant in the final model. The Monte Carlo samples from the posterior distributions for the parameter coefficients are shown in Figure 5.4. These were for the model fit with constant  $g = N$ . The samples from the Zellner-Siow approach were very similar and hence omitted. The values above

the spikes in the histograms represent the probabilities of assigning that variable to zero (these are equivalent to 1 minus the probabilities in Table 5.1 above). The resulting coefficient estimates and 95% credible intervals are given in Table 5.2.



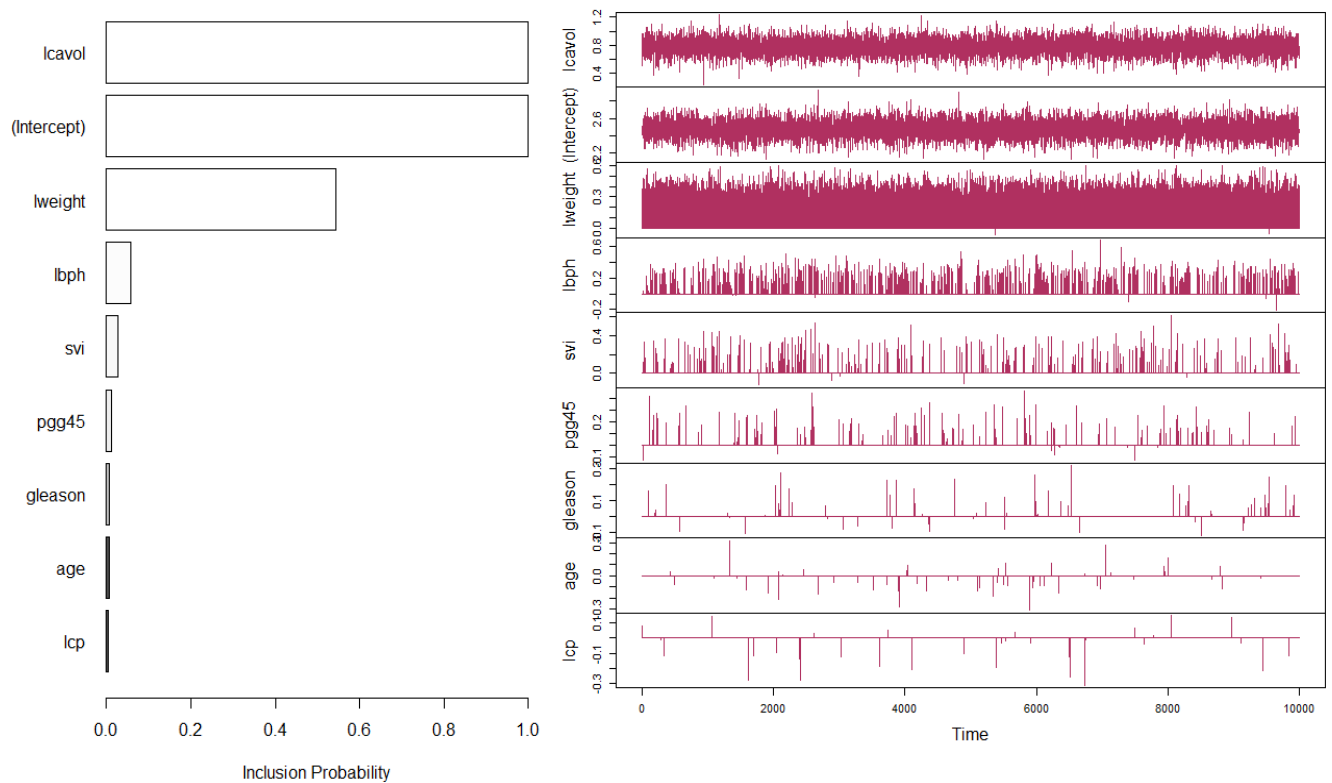
**Figure 5.4: Approximate posterior distributions obtained from MCMC.**

**Table 5.2: Posterior medians and 95% credible intervals for each parameter using the two variants of Zellner's  $g$ -prior.**

	<i>g</i> -Prior (constant)		<i>g</i> -Prior (Zellner-Siow)	
	Posterior Median	95% Credible Interval	Posterior Median	95% Credible Interval
<b>Intercept</b>	2.4715	(2.2873; 2.6573)	2.4713	(2.2870; 2.6513)
<b>lcavol</b>	0.6657	(0.4083; 0.9134)	0.6653	(0.4061; 0.9152)
<b>lweight</b>	0.2820	(0.0000; 0.4843)	0.2778	(0.0000; 0.4812)
<b>age</b>	0.0000	(-0.2063; 0.0087)	0.0000	(-0.2262; 0.0235)
<b>lbph</b>	0.0000	(0.0000; 0.3978)	0.0000	(0.0000; 0.3966)
<b>svi</b>	0.08455	(0.0000; 0.4786)	0.1186	(0.0000; 0.4879)
<b>lcp</b>	0.0000	(-0.3587; 0.0293)	0.0000	(-0.3948; 0.0275)
<b>gleason</b>	0.0000	(-0.0690; 0.1699)	0.0000	(-0.1056; 0.1862)
<b>pgg45</b>	0.0000	(0.0000; 0.3499)	0.0000	(0.0000; 0.3681)

The second method of variable selection was the use of a spike-and-slab prior according to the method of George & McCulloch (1993). Here, an absolutely continuous spike was used where a conditional Gaussian prior was defined with a precision matrix equal to the amount of information contained in a single point by using Zellner's unit information prior. The inclusion probabilities are displayed graphically in the left plot of Figure 5.5 with the corresponding trace plots for the parameters coefficients given on the right of Figure 5.5





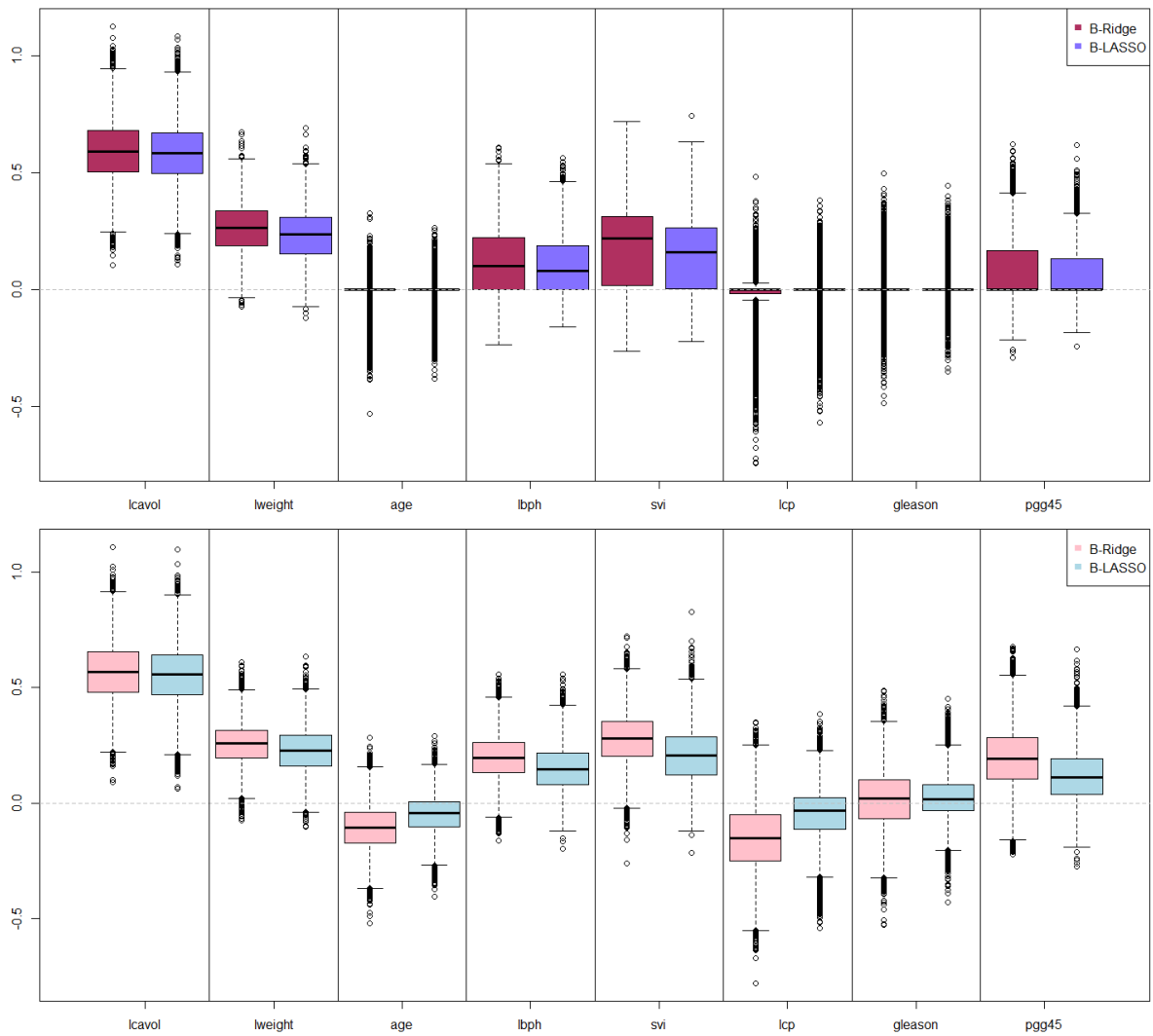
**Figure 5.5: Inclusion probabilities (left) and trace plots (right) for each parameter using a spike-and-slab prior.**

According to these inclusion probabilities, only 2 variables (in addition to the intercept) are considered significant in the model. The variable ‘svi’ (seminal vesicle invasion) is no longer included in the model when using a spike-and-slab prior, whereas it was included when Zellner’s  $g$ -prior was used and so the spike-and-slab prior has consequently selected a sparser model. The posterior medians and credible intervals are given in Table 5.3.

**Table 5.3: Posterior median and 95% credible interval for each parameter using a spike-and-slab prior.**

	<b>Posterior Median</b>	<b>95% Credible Interval</b>
<b>Intercept</b>	2.4777	(2.2864; 2.6641)
<b>lcavol</b>	0.7788	(0.5639; 0.9892)
<b>lweight</b>	0.1958	(0.0000; 0.4632)
<b>age</b>	0.0000	(0.0000; 0.0000)
<b>lbph</b>	0.0000	(0.0000; 0.2780)
<b>svi</b>	0.0000	(0.0000; 0.0076)
<b>lcp</b>	0.0000	(0.0000;0.0000)
<b>gleason</b>	0.0000	(0.0000;0.0000)
<b>pgg45</b>	0.0000	(0.0000;0.0000)

Finally, Bayesian ridge regression and the Bayesian LASSO were performed on the training data. This was done in two separate ways: (1) including reversible jump (RJ) MCMC (which was introduced in Section 2.2.2.3), and (2) without reversible jump MCMC. Since the Bayesian ridge regression and LASSO do not shrink coefficients exactly to zero, the RJ mechanisms performs this variable selection and removes some predictors entirely from the model. The results of the parameter estimates sampled from their posterior distribution for each of these two runs are displayed via the use of box plots in Figure 5.6. Not that the intercept has not been included for aesthetic purposes. In both cases, the LASSO estimates are shrunk closer to zero and tend to have a smaller interquartile range corresponding to a smaller posterior variance. The RJ mechanisms removes the variables 'age', 'lcp', 'gleason' and 'pgg45' entirely from the model. The resulting posterior medians and 95% credible intervals for each of these models and their predictors is given in Table 5.4 and Table 5.5.



**Figure 5.6: Box plots of the posterior distributions for the Bayesian LASSO and Bayesian ridge regression using reversible jump MCMC (top) and without reversible jump MCMC (bottom).**

**Table 5.4: Posterior medians and 95% credible intervals for Bayesian ridge regression with and without reversible jump MCMC.**

	Bayesian Ridge Regression (with RJ)		Bayesian Ridge Regression	
	Posterior Median	95% Credible Interval	Posterior Median	95% Credible Interval
<b>Intercept</b>	2.4680	(2.2831; 2.6469)	2.4690	(2.2931; 2.6385)
<b>lcavol</b>	0.5891	(0.4767; 0.7407)	0.5662	(0.4505; 0.7362)
<b>lweight</b>	0.2642	(0.1160; 0.3505)	0.2572	(0.1196; 0.2442)
<b>age</b>	0.0000	(0.0000; 0.0000)	-0.1082	(-0.1779; 0.0918)
<b>lbph</b>	0.1013	(0.1487; 0.1661)	0.1965	(0.0497; 0.0539)
<b>svi</b>	0.2162	(0.0227; 0.3160)	0.2786	(0.2143; 0.2651)
<b>lcp</b>	0.0000	(0.0000; 0.0000)	-0.1512	(-0.2799; -0.2219)
<b>gleason</b>	0.0000	(-0.1097; -0.0028)	0.0190	(-0.1232; -0.0917)
<b>pgg45</b>	0.0000	(0.0000; 0.0000)	0.1921	(0.4157; 0.4671)

**Table 5.5: Posterior medians and 95% credible intervals for the Bayesian LASSO with and without reversible jump MCMC.**

	Bayesian LASSO (with RJ)		Bayesian LASSO	
	Posterior Median	95% Credible Interval	Posterior Median	95% Credible Interval
<b>Intercept</b>	2.4680	(2.2781; 2.6568)	2.4680	(2.2789; 2.6571)
<b>lcavol</b>	0.5809	(0.4195; 0.6920)	0.5565	(0.4839; 0.5873)
<b>lweight</b>	0.2358	(0.3295; 0.3438)	0.2266	(0.2432; 0.2661)
<b>age</b>	0.0000	(0.0000; 0.0000)	-0.0429	(-0.1446; 0.0131)
<b>lbph</b>	0.0780	(0.0000; 0.0000)	0.1457	(0.1192; 0.2503)
<b>svi</b>	0.1578	(0.0033; 0.1298)	0.2052	(0.2871; 0.3598)
<b>lcp</b>	0.0000	(-0.0227; 0.0677)	-0.0345	(-0.1798; -0.1567)
<b>gleason</b>	0.0000	(-0.0640; -0.0016)	0.0175	(-0.0743; 0.0881)
<b>pgg45</b>	0.0000	(0.0525; 0.1757)	0.1093	(0.1133; 0.2332)

Each of these models were then applied to the held-out test set and their respective test errors and estimated standard errors were obtained. The median of each of the posterior distributions for the parameters was used as a point estimate for this calculation. The resulting test error and estimated standard error for each model is given in Table 5.6. The final column in Table 5.6 indicates the number of variables in the model, not including the intercept term. The 'Full Model' corresponds to the classical linear regression model fit to the entire dataset prior to any variable selection.

**Table 5.6: Comparison of test error, standard error and number of variables in the final model for each of the variable selection techniques.**

Model	Test Error	Standard Error	No. of variables
<b>Full Model</b>	0.5213	0.1787	8
<i>g</i> -Prior (constant)	0.4449	0.1387	3
<i>g</i> -Prior (Zellner-Siow)	0.4480	0.1407	3
<b>Spike-and-Slab</b>	0.4733	0.1363	2
<b>Bayesian Ridge (RJ)</b>	0.4323	0.1278	4
<b>Bayesian Ridge</b>	0.4920	0.1626	Shrinkage
<b>Bayesian LASSO (RJ)</b>	0.4465	0.1407	4
<b>Bayesian LASSO</b>	0.4564	0.1483	Shrinkage

Overall, the Bayesian ridge regression with reversible jump MCMC resulted in the lowest test error. However, if the two reversible jump methods are excluded, then assigning a *g*-Prior, specifically using the constant value of  $g = N$ , outperformed the other methods and found only a subset of 3 variables was necessary in the model. Although the spike-and-slab prior resulted in the sparsest model with only 2 variables, it had a higher test error than some of the other models. In total, all models resulted in a lower test error than the full model fit using classical linear regression.

## 5.6 THE RELEVANCE VECTOR MACHINE

As with most machine learning techniques, support vector machines (SVMs), despite being very useful and powerful, have their drawbacks. Some of the most notable being that the SVM only returns class memberships and not posterior probabilities (i.e. we cannot know how well a particular point is classified); the training time for large datasets can often be very long (also

because we need to determine the value of the cost complexity parameter using methods such as cross-validation) and the kernel functions in the formula of the SVM are also required to be positive definite.

The relevance vector machine (RVM), introduced by Tipping (2001), was proposed as a method to maintain the advantageous properties of the support vector machines whilst attempting to provide solutions to its major drawbacks. Like the SVM, the RVM is also a sparse kernel method and can be used in both regression and classification settings. The RVM makes the assumption that not all the basis functions will be necessary in the model. When working with models that are in the form of linear combinations of the inputs, it is often the case that a certain combination of some of these basis functions in place of others can produce the outputs without a significant reduction in the accuracy of the results.

The major advantage of the RVM over the SVM is that it tends to produce much sparser models and so it is computationally faster to apply to test sets without having a negative effect on the generalization error. Having sparser models means that our solution is more parsimonious and thus easier to work with and interpret (Bishop, 2006:345).

### 5.6.1 Support Vector Machines

Support vector machines are a common machine learning technique that can be used in both classification and regression scenarios. Originally designed for classification problems, the form of the support machine arises by kernelizing the support vector classifier (SVC). A comprehensive overview of support vector machines is beyond the scope of this thesis, however, a brief summary of the most important aspects of support vector machines will be provided as context for the comparison of support vector machines and relevance vector machines. A complete explanation of support vector machines can be found in Hastie *et al.* (2009: 129–135, 417–438).

We consider the simple case of 2-dimensional, binary data for explanation and illustration purposes. Hence, consider a training set  $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$  with response classes  $y_n \in \{-1, 1\}$  for all  $n$ . If the data are linearly separable, it is possible to construct an optimal separating hyperplane (OSH) that best separates the data into two classes without any overlap of points. Further, the OSH constructs a hard margin around the decision boundary in which no training points can fall. Hence, the aim of the OSH is to obtain a unique decision boundary by constructing a margin that separates the closest point from each class as far as possible. However, this does not necessarily mean that a new test data point will not fall into this margin or will even be correctly classified at all based on this hyperplane.

The SVC then extends this concept and is constructed such that the margin of separation between the two classes is set to be as large as possible, with some allowance for overlap of points into the margin and even over the decision boundary to avoid overfitting. In other words, the SVC

constructs a soft margin around the decision boundary. This overlap is controlled via the use of a cost complexity parameter that is usually determined using cross-validation.

The decision boundary for an SVC is given by

$$\hat{f}(\mathbf{x}) = \hat{\mathbf{w}}^T \mathbf{x} + \hat{w}_0 \quad (5.57)$$

where the parameter estimates satisfy

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (5.58)$$

$$\text{subject to } \xi_i \geq 0, \quad y_i(\mathbf{x}_i^T \mathbf{w} + w_0) \geq 1 - \xi_i, \quad \forall_i$$

where  $\xi_i, i = 1, \dots, N$  denote the slack variables which control the total amount of overlap into the margin as well as over the decision boundary into the other class and  $C$  is the cost complexity parameter (Hastie *et al.*, 2009:424). The parameters  $\mathbf{w} = [w_0, \dots, w_N]^T$  are the weights obtained by maximizing the separating margin between the two classes and  $w_0$  represents the bias term in the model.

This decision boundary obtained by the SVC is a linear decision boundary. The true boundary between two classes is very seldom linear and this prompted the introduction of the support vector machine. The SVM is a kernelized version of the SVC which results in a non-linear decision boundary in the original predictor variables. However, an important point to note is that even though this decision boundary is non-linear in the original predictor variables, it is a linear decision boundary in the space of the transformed predictor variables.

The support vector machine thus is trained according to the model

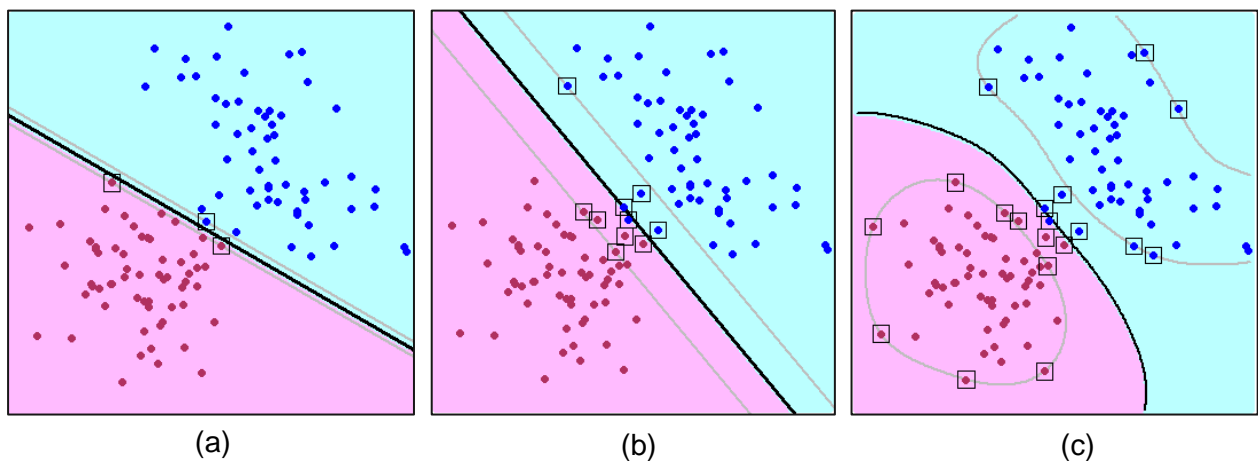
$$f(\mathbf{x}) = \sum_{i=1}^N w_i y_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i) + w_0 \quad (5.59)$$

where  $\mathcal{K}$  denotes the kernel function. Note that there is no distributional assumption made on any of the elements in the model given above. Some popular choices for kernel functions in the literature are given in Table 5.7.

**Table 5.7: Popular kernel functions for support vector machines.**

Kernel	Function
$d$ th-degree polynomial	$(1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^d$
Radial Basis	$\exp(-\gamma \ \mathbf{x} - \mathbf{x}'\ ^2)$
Neural network	$\tanh(\kappa_1 \langle \mathbf{x}, \mathbf{x}' \rangle + \kappa_2)$

Figure 5.7 provides a simple illustration of the difference between an OSH, SVC and SVM. Since the data are linearly separable, an optimal separating hyperplane exists, as given in Figure 5.7(a), and corresponds to a support vector classifier with an infinitely large cost parameter which prevents any points from falling into the margin or over the boundary. Clearly, this could lead to an overfit decision boundary. Figure 5.7(b) displays a support vector classifier where points are now allowed to lie inside the margin, and some are even sitting on the wrong side of the decision boundary. This is clearly a more flexible fit that should most probably lead to a lower test error, even though it would have a larger training error than the OSH. Finally, Figure 5.7(c) represents a support vector machine with a radial basis kernel function. This boundary is no longer linear but is more now flexible, albeit more complex.



**Figure 5.7: Optimal separating hyperplane (a), support vector classifier (b) and support vector classifier (c) for the simulated linearly separable data.**

The data points in Figure 5.7 that have a square plotted around them are presenting which of the training data are the support vectors. The support vectors are the only points which actually contribute to the construction of the decision boundary and the margin and are naturally those points which lie closest to the decision boundary and are the most difficult points to classify. These support vectors are the data points which correspond to the kernel functions containing non-zero coefficients. Hence, the decision boundary is determined only using a subset of the training data and this subset is the set of support vectors.

Although not discussed here, support vector machines can easily be adapted to be applied in regression settings. However, support vector machines do tend to be more commonly encountered in classification rather than regression settings. This is in direct contrast to the relevance vector machine discussed in the next section. A discussion of support vector regression can be found in Hastie *et al.* (2009: 434-436).



### 5.6.2 Relevance Vector Regression

The relevance vector machine (Tipping, 2001) takes on a similar functional form to the SVM: it is a linear model which contains kernel functions. The difference now is that it includes a specific prior distribution over the parameters that leads to the desired sparsity. It is essentially the probabilistic interpretation of the support vector machine.

In the linear regression scenario of Chapter 4, the conditional distribution for the response variable was modelled by

$$p(\mathbf{x}, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(\mathbf{x}^T \boldsymbol{\theta}, \tau^{-1}). \quad (5.60)$$

As mentioned, we can extend our linear model to be in the form of a basis expansion, using basis functions  $\phi_i(\mathbf{x}), i = 1, \dots, M$  without losing any of the properties or results that have been derived. Thus, the conditional distribution for the model

$$y = \sum_{i=1}^M \theta_i \phi_i(\mathbf{x}) + \varepsilon = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) + \varepsilon = f(\mathbf{x}) + \varepsilon \quad (5.61)$$

where  $\boldsymbol{\phi}(\mathbf{x})$  is an  $M$ -dimensional vector of basis functions, is then given by

$$p(y|\mathbf{x}, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(f(\mathbf{x}), \tau^{-1}) \quad (5.62)$$

through immediate extension of the previous result in Equation 5.60.

Comparable to the structure of the SVM, the RVM also makes use of a kernelized form of the linear regression model where the basis functions are now replaced by kernels, with one kernel for each data point. Hence, it has the form

$$f(\mathbf{x}) = \sum_{n=1}^N \theta_n \mathcal{K}(\mathbf{x}, \mathbf{x}_n) + \theta_0 \quad (5.63)$$

where  $\theta_0$  denotes the bias parameter. This has the same form as the SVM, but already one advantage arises since the kernels are not restricted to being positive definite and the basis functions are not tied in either number or location to the training points.

Given our training dataset  $\mathcal{D} = \{(\mathbf{y}_n, \mathbf{x}_n), n = 1, \dots, N\}$ , the likelihood function for this data is

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau) = \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\theta}; \tau) \sim \mathcal{N}(\boldsymbol{\Phi} \boldsymbol{\theta}, \tau^{-1} \mathbf{I}) \quad (5.64)$$

where  $\boldsymbol{\Phi}: N \times (N + 1)$  denotes the design matrix of basis functions, with entries  $[\boldsymbol{\Phi}]_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$  with the first column a column of ones for the intercept term.

The prior distribution assigned to the parameters for the RVM is a zero mean Gaussian prior and we further introduce a separate hyperparameter,  $\alpha_i^{-1}$ , for each of the parameters  $\theta_i$ . This is how

the sparsity will be induced and is in contrast to the linear regression scenario in Chapter 4 where we worked with a single shared hyperparameter  $\alpha^{-1}$  for the precision. Hence, the prior distribution is given by

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \prod_{i=1}^M \mathcal{N}(0, \alpha_i^{-1}) \quad (5.65)$$

where  $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_M]^T$ .

The reason that the RVM induces sparsity into the model is because when the evidence function is maximized with respect to the hyperparameters, many of them tend towards infinity which results in the corresponding weight parameters being forced towards zero. This means that these corresponding basis functions play no significant role in the model and can hence be excluded, thus leading to model sparsity.

The posterior distribution for the parameters is given by

$$p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}; \boldsymbol{\alpha}, \tau) \propto p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau) p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \quad (5.66)$$

which is directly found to be

$$p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}; \boldsymbol{\alpha}, \tau) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (5.67)$$

with mean and covariance matrix

$$\boldsymbol{\mu} = \tau \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y} \quad (5.68)$$

$$\boldsymbol{\Sigma} = (\mathbf{A} + \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}. \quad (5.69)$$

Here, the matrix  $\mathbf{A}$  is a diagonal matrix with precision parameters on the diagonal, that is,  $\mathbf{A} = \text{diag}(\alpha_i)$ .

The values of the hyperparameters  $\boldsymbol{\alpha}$  and  $\tau$  can be determined using Type-II maximum likelihood, that is, we integrate out the weight parameters to obtain the marginal likelihood function to be maximized:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}; \tau) p(\boldsymbol{\theta}|\boldsymbol{\alpha}) d\boldsymbol{\theta} \\ &= \int \left\{ \left( \frac{\tau}{2\pi} \right)^{\frac{N}{2}} \exp \left( -\frac{\tau}{2} (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\theta})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\theta}) \right) \right\} \left\{ (2\pi)^{-\frac{M}{2}} \left( \prod_{i=1}^M \alpha_i \right) \exp \left( -\frac{1}{2} \boldsymbol{\theta}^T \mathbf{A} \boldsymbol{\theta} \right) \right\} d\boldsymbol{\theta} \\ &= \left( \frac{\tau}{2\pi} \right)^{\frac{N}{2}} (2\pi)^{-\frac{M}{2}} \left( \prod_{i=1}^M \alpha_i \right) \int \exp(-g(\boldsymbol{\theta})) d\boldsymbol{\theta} \end{aligned} \quad (5.70)$$

where

$$g(\boldsymbol{\theta}) = \frac{\tau}{2} (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\theta})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\theta}) + \frac{1}{2} \boldsymbol{\theta}^T \mathbf{A} \boldsymbol{\theta}. \quad (5.71)$$

As before, we can complete the square and isolate all terms containing  $\boldsymbol{\theta}$  to obtain

$$g(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}) + h(\mathbf{y}) \quad (5.72)$$

with  $h(\mathbf{y}) = \frac{1}{2}(\tau \mathbf{y}^T \mathbf{y} - \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu})$ .

This means that our integral evaluates as

$$\int \exp(-g(\boldsymbol{\theta})) d\boldsymbol{\theta} = (2\pi)^{\frac{M}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}} \exp(-h(\mathbf{y})). \quad (5.73)$$

Hence, we have

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= \left(\frac{\tau}{2\pi}\right)^{\frac{N}{2}} (2\pi)^{-\frac{M}{2}} \left(\prod_{i=1}^M \alpha_i\right) \int \exp(-g(\boldsymbol{\theta})) d\boldsymbol{\theta} \\ &= \left(\frac{\tau}{2\pi}\right)^{\frac{N}{2}} (2\pi)^{-\frac{M}{2}} \left(\prod_{i=1}^M \alpha_i\right) (2\pi)^{\frac{M}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}} \exp(-h(\mathbf{y})). \end{aligned} \quad (5.74)$$

Since we are interested in the variable  $\mathbf{y}$ , we only need to consider the term  $\exp(-h(\mathbf{y}))$  because the remaining terms are constants with respect to  $\mathbf{y}$ . In the derivations to follow, we repeatedly make use of the fact that  $\boldsymbol{\mu} = \tau \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y}$  and  $\boldsymbol{\Sigma} = (\mathbf{A} + \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}$  as well as the Woodbury matrix identity. The Woodbury matrix identity, which is a matrix generalization of the Sherman-Morrison formula (Sherman & Morrison, 1950) states that the following matrix inverse holds:

$$(\mathbf{A} + \mathbf{B} \mathbf{D}^{-1} \mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} (\mathbf{D} + \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{C} \mathbf{A}^{-1} \quad (5.75)$$

where  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  and  $\mathbf{D}$  are the relevant invertible matrices of required conforming sizes.

Hence,

$$\begin{aligned} h(\mathbf{y}) &= \frac{1}{2}(\tau \mathbf{y}^T \mathbf{y} - \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) \\ &= \frac{1}{2}(\tau \mathbf{y}^T \mathbf{y} - \tau \mathbf{y}^T \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \tau \mathbf{y}) \\ &= \frac{1}{2} \mathbf{y}^T (\tau \mathbf{I} - \tau \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \tau) \mathbf{y} \\ &= \frac{1}{2} \mathbf{y}^T (\tau \mathbf{I} - \tau \boldsymbol{\Phi} (\mathbf{A} + \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \tau) \mathbf{y} \\ &= \frac{1}{2} \mathbf{y}^T (\tau \mathbf{I} - \tau \mathbf{I} \boldsymbol{\Phi} (\mathbf{A} + \boldsymbol{\Phi}^T \tau \mathbf{I} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \tau \mathbf{I}) \mathbf{y}, \quad \text{from the Woodbury identity} \\ &= \frac{1}{2} \mathbf{y}^T (\tau^{-1} \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^T)^{-1} \mathbf{y} \\ &= \frac{1}{2} \mathbf{y}^T \mathbf{C}^{-1} \mathbf{y} \end{aligned} \quad (5.76)$$

where  $\mathbf{C} = \tau^{-1}\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^T$ . Hence,

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &\propto \exp(-h(\mathbf{y})) \\ &= \exp\left(-\frac{1}{2}\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y}\right). \end{aligned} \quad (5.77)$$

Thus, the marginal likelihood function is a zero-mean Gaussian distribution with covariance matrix  $\mathbf{C}$ , and so the log marginal likelihood can be expressed as

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= \log\left\{(2\pi)^{-\frac{N}{2}}|\mathbf{C}|^{-\frac{1}{2}}\exp(\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y})\right\} \\ &= -\frac{1}{2}\{N\log 2\pi + \log|\mathbf{C}| + \mathbf{y}^T\mathbf{C}^{-1}\mathbf{y}\}. \end{aligned} \quad (5.78)$$

We now wish to maximize this quantity with respect to the hyperparameters  $\boldsymbol{\alpha}$  and  $\tau$ . This can be done directly by taking the derivative of Equation 5.78, setting that derivative equal to zero and subsequently solving for the respective parameters. Alternatively, we can make use of the EM algorithm. Both methods will arrive at the same result, but, in this case, direct optimization via derivatives is computationally faster.

We firstly rewrite the log-likelihood in a more useable form:

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= -\frac{1}{2}\{N\log 2\pi + \log|\mathbf{C}| + \mathbf{y}^T\mathbf{C}^{-1}\mathbf{y}\} \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{C}| - \frac{1}{2}\log \mathbf{y}^T\mathbf{C}^{-1}\mathbf{y} \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\tau^{-1}\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^T| - \frac{1}{2}\log \mathbf{y}^T(\tau^{-1}\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^T)^{-1}\mathbf{y}. \end{aligned} \quad (5.79)$$

This expression can be simplified by using the following matrix determinant lemma

$$\begin{aligned} |\mathbf{A}||\tau^{-1}\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^T| &= |\tau^{-1}\mathbf{I}||\mathbf{A} + \tau\Phi^T\Phi| \\ \Rightarrow |\sigma^2\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^T| &= \frac{|\tau^{-1}\mathbf{I}||\mathbf{A} + \tau\Phi^T\Phi|}{|\mathbf{A}|}. \end{aligned} \quad (5.80)$$

Hence, we can then write

$$\begin{aligned} -\frac{1}{2}\log|\tau^{-1}\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^T| &= -\frac{1}{2}\log\left\{\frac{|\tau^{-1}\mathbf{I}||\mathbf{A} + \tau\Phi^T\Phi|}{|\mathbf{A}|}\right\} \\ &= -\frac{1}{2}\{\log|\tau^{-1}\mathbf{I}| + \log|\mathbf{A} + \tau\Phi^T\Phi| - \log|\mathbf{A}|\} \\ &= -\frac{1}{2}\log \tau^{-N} + \frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\log|\mathbf{A}| \\ &= \frac{1}{2}\sum_{i=0}^N \log \alpha_i + \frac{N}{2}\log \tau + \frac{1}{2}\log|\boldsymbol{\Sigma}|. \end{aligned} \quad (5.81)$$

Further, we simplify the final term of the log marginal likelihood function by again using the Woodbury identity and the fact that  $\boldsymbol{\mu} = \tau \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y}$  and  $\boldsymbol{\Sigma} = (\mathbf{A} + \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}$ . Thus,

$$\begin{aligned}
 -\frac{1}{2} \log \mathbf{y}^T (\tau^{-1} \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^T)^{-1} \mathbf{y} &= -\frac{1}{2} \mathbf{y}^T (\tau \mathbf{I} - \tau \boldsymbol{\Phi} (\mathbf{A} + \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \tau) \mathbf{y} \\
 &= -\frac{\tau}{2} [\mathbf{y}^T \mathbf{y} - \tau \mathbf{y}^T \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y}] \\
 &= -\frac{\tau}{2} [\mathbf{y}^T \mathbf{y} - \mathbf{y}^T \boldsymbol{\Phi} \boldsymbol{\mu}] \\
 &= -\frac{\tau}{2} [(\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu}) + \mathbf{y}^T \boldsymbol{\Phi} \boldsymbol{\mu} - \boldsymbol{\mu}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{\mu}] \\
 &= -\frac{\tau}{2} [(\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu}) + \beta^{-1} \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - \boldsymbol{\mu}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{\mu}] \\
 &= -\frac{1}{2} [\tau (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu}) + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - \tau \boldsymbol{\mu}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{\mu}] \\
 &= -\frac{1}{2} [\tau (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu}) + \boldsymbol{\mu}^T (\boldsymbol{\Sigma}^{-1} - \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi}) \boldsymbol{\mu}] \\
 &= -\frac{1}{2} [\tau (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu}) + \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu}]. \tag{5.82}
 \end{aligned}$$

Hence,

$$\begin{aligned}
 \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= -\frac{N}{2} \log 2\pi - \frac{1}{2} \log |\tau^{-1} \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^T| - \frac{1}{2} \log \mathbf{y}^T (\tau^{-1} \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^T)^{-1} \mathbf{y} \\
 &= -\frac{N}{2} \log 2\pi + \frac{1}{2} \sum_{i=0}^N \log \alpha_i + \frac{N}{2} \log \tau + \frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} [\tau (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi} \boldsymbol{\mu}) + \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu}]. \tag{5.83}
 \end{aligned}$$

Thus, taking the derivative of 5.83 firstly with respect to  $\alpha_i$  and setting equal to zero gives:

$$\frac{\partial}{\partial \alpha_i} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) = \frac{1}{2\alpha_i} + \frac{1}{2} \frac{\partial}{\partial \alpha_i} \log |\boldsymbol{\Sigma}| - \frac{1}{2} \frac{\partial}{\partial \alpha_i} \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu}. \tag{5.84}$$

Now,

$$\frac{\partial}{\partial \alpha_i} \log |\boldsymbol{\Sigma}| = -\frac{\partial}{\partial \alpha_i} \log |\boldsymbol{\Sigma}^{-1}| = -\text{tr} \left\{ \boldsymbol{\Sigma} \frac{\partial \boldsymbol{\Sigma}^{-1}}{\partial \alpha_i} \right\} = -\text{tr} \left\{ \boldsymbol{\Sigma} \frac{\partial \mathbf{A}}{\partial \alpha_i} \right\} = -\boldsymbol{\Sigma}_{ii} \tag{5.85}$$

and

$$\frac{\partial}{\partial \alpha_i} \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu} = \mu_i^2. \tag{5.86}$$

Hence,

$$\begin{aligned}
 \frac{\partial}{\partial \alpha_i} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= \frac{1}{2\alpha_i} - \frac{1}{2} \boldsymbol{\Sigma}_{ii} - \frac{1}{2} \mu_i^2 = 0 \\
 \Rightarrow \frac{1}{2\hat{\alpha}_i} - \frac{1}{2} \boldsymbol{\Sigma}_{ii} - \frac{1}{2} \mu_i^2 &= 0
 \end{aligned}$$

$$\begin{aligned}
 &\Rightarrow \frac{1}{\hat{\alpha}_i} - \Sigma_{ii} = \mu_i^2 \\
 &\Rightarrow \frac{1 - \hat{\alpha}_i \Sigma_{ii}}{\hat{\alpha}_i} = \mu_i^2 \\
 &\Rightarrow \hat{\alpha}_i \mu_i^2 = 1 - \hat{\alpha}_i \Sigma_{ii} \\
 &\Rightarrow \hat{\alpha}_i = \frac{1 - \hat{\alpha}_i \Sigma_{ii}}{\mu_i^2} \\
 &\Rightarrow \hat{\alpha}_i = \frac{\gamma_i}{\mu_i^2}
 \end{aligned} \tag{5.87}$$

where  $\gamma_i = 1 - \alpha_i \Sigma_{ii}$ .

Similarly, to obtain the estimate for  $\tau$ , we again take the derivative of 5.83 with respect to  $\tau$  and set equal to zero and solve:

$$\frac{\partial}{\partial \tau} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) = \frac{N}{2\tau} + \frac{1}{2} \frac{\partial}{\partial \tau} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}) - \frac{1}{2} \frac{\partial}{\partial \tau} \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu}. \tag{5.88}$$

Now,

$$\frac{\partial}{\partial \tau} \log |\boldsymbol{\Sigma}| = -\frac{\partial}{\partial \tau} \log |\boldsymbol{\Sigma}^{-1}| = -\text{tr} \left\{ \boldsymbol{\Sigma} \frac{\partial \boldsymbol{\Sigma}^{-1}}{\partial \tau} \right\} = -\text{tr} \left\{ \boldsymbol{\Sigma} \frac{\partial \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi}}{\partial \tau} \right\} = -\text{tr} \{ \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{\Phi} \}. \tag{5.89}$$

To isolate the terms involving  $\tau$  for the differentiation, we rewrite the matrices inside the trace of Equation 5.89 in the following equivalent form:

$$\begin{aligned}
 \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{\Phi} &= \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \tau^{-1} \boldsymbol{\Sigma} \mathbf{A} - \tau^{-1} \boldsymbol{\Sigma} \mathbf{A} \\
 &= \boldsymbol{\Sigma} (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \tau^{-1} \mathbf{A}) - \tau^{-1} \boldsymbol{\Sigma} \mathbf{A} \\
 &= \boldsymbol{\Sigma} (\tau \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mathbf{A}) \tau^{-1} - \tau^{-1} \boldsymbol{\Sigma} \mathbf{A} \\
 &= (\mathbf{A} + \tau \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} (\tau \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mathbf{A}) \tau^{-1} - \tau^{-1} \boldsymbol{\Sigma} \mathbf{A} \\
 &= \tau^{-1} - \tau^{-1} \boldsymbol{\Sigma} \mathbf{A} \\
 &= \tau^{-1} (\mathbf{I} - \boldsymbol{\Sigma} \mathbf{A}).
 \end{aligned} \tag{5.90}$$

Hence,

$$\frac{\partial}{\partial \tau} \log |\boldsymbol{\Sigma}| = -\text{tr} \{ \tau^{-1} (\mathbf{I} - \boldsymbol{\Sigma} \mathbf{A}) \} = -\frac{1}{\tau} \sum_i (1 - \alpha_i \Sigma_{ii}) = -\frac{1}{\tau} \sum_i \gamma_i. \tag{5.91}$$

And so, we have

$$\begin{aligned}
 \frac{\partial}{\partial \tau} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \tau) &= \frac{N}{2\tau} + \frac{1}{2} \frac{\partial}{\partial \tau} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}) - \frac{1}{2} \frac{\partial}{\partial \tau} \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu} \\
 &\Rightarrow \frac{N}{2\hat{\tau}} - \frac{\sum_i \gamma_i}{2\hat{\tau}} - \frac{1}{2} (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}) = 0 \\
 &\Rightarrow \frac{N - \sum_i \gamma_i}{\hat{\tau}} = (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu})
 \end{aligned}$$

$$\Rightarrow \hat{\tau} = \frac{N - \sum_i \gamma_i}{(\mathbf{y} - \Phi \boldsymbol{\mu})^T (\mathbf{y} - \Phi \boldsymbol{\mu})}. \quad (5.92)$$

Thus, the estimates for the hyperparameters are given by

$$\hat{\alpha}_i = \frac{\gamma_i}{\mu_i^2} \quad (5.93)$$

$$\hat{\tau}^{-1} = \frac{(\mathbf{y} - \Phi \boldsymbol{\mu})^T (\mathbf{y} - \Phi \boldsymbol{\mu})}{N - \sum_i \gamma_i} \quad (5.94)$$

where

$$\gamma_i = 1 - \alpha_i \Sigma_{ii} \quad (5.95)$$

and  $\Sigma_{ii}$  denotes the  $i$ th diagonal element of  $\Sigma$ . This provides a measure of how well the parameter  $\theta_i$  is determined by the data. We immediately notice that both solutions are implicit and so an iterative procedure is required to obtain the final estimates of these values. Hence, the learning proceeds by selecting initial starting values for  $\boldsymbol{\alpha}$  and  $\tau$  and evaluating the posterior mean and covariance matrix, and then using these to update the estimates for the hyperparameters. These new hyperparameters then update the estimates for the parameters of the posterior distribution and the process evolves iteratively in that manner.

Upon convergence, a significant proportion of the parameters will be found to have diverged to infinity, corresponding to parameter estimates with a posterior distribution having mean and covariance of zero. Hence, the basis functions corresponding to these parameters can be removed from the model since they do not statistically significantly influence the model. Similarly to the role played by support vectors with SVMs, the datapoints  $\mathbf{x}_n$  that correspond to non-zero parameter values are known as the relevance vectors, their name deriving from the fact that they have been obtained through this above process which has been termed automatic relevance determination (Neal, 1996; MacKay, cited in Tipping, 2001); a process that was initially developed in the framework of neural networks.

Once the final parameter and hyperparameter estimates have been obtained, we are able to compute the predictive distribution for a new data observation  $\mathbf{x}$ . This is defined by

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}; \hat{\boldsymbol{\alpha}}, \hat{\tau}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}; \hat{\tau}) p(\boldsymbol{\theta}|\mathcal{D}; \hat{\boldsymbol{\alpha}}, \hat{\tau}) d\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}^T \boldsymbol{\phi}(\mathbf{x}), \sigma_x^2) \quad (5.96)$$

where

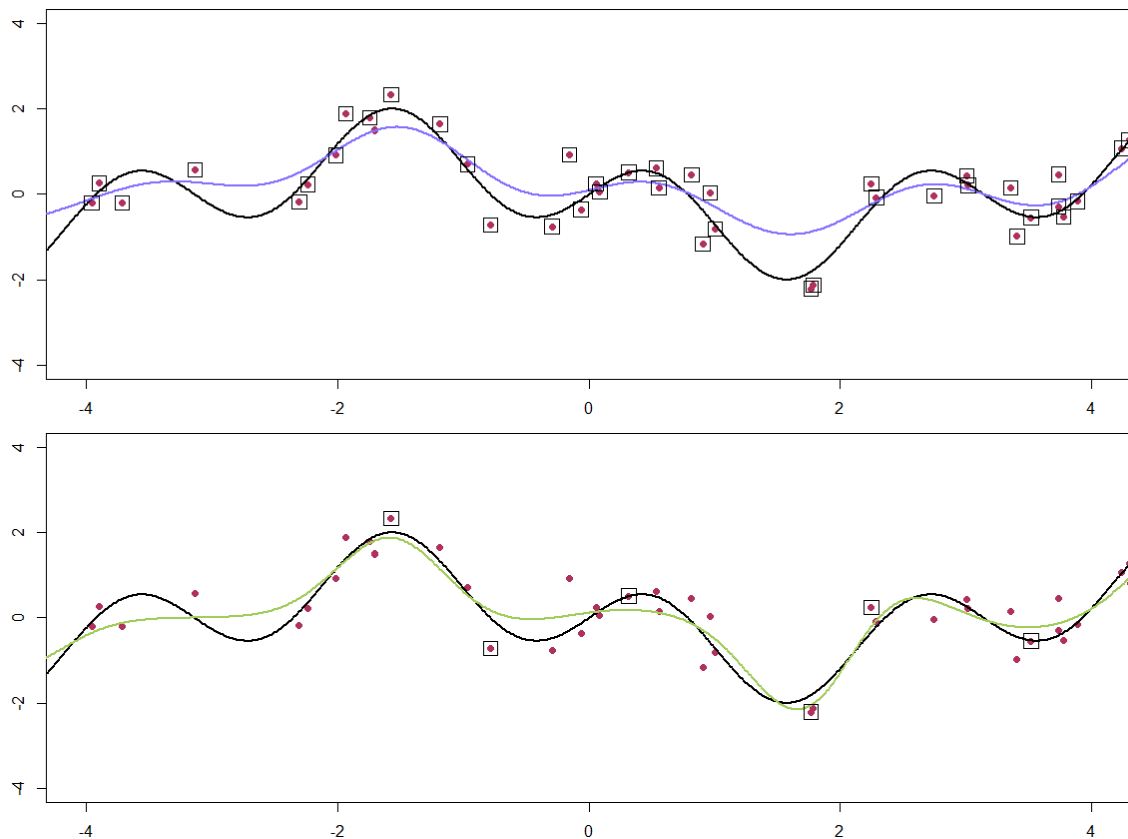
$$\sigma_x^2 = \hat{\tau}^{-1} + \boldsymbol{\phi}(\mathbf{x})^T \Sigma \boldsymbol{\phi}(\mathbf{x}) \quad (5.97)$$

with  $\boldsymbol{\mu}$  and  $\Sigma$  defined as previously with  $\boldsymbol{\alpha}$  and  $\tau$  replaced by their respective estimates.

To demonstrate the performance of relevance vector regression, we consider data simulated from the function

$$y = \sin(3x) + \sin(-x) \quad (5.98)$$

where a sample of size  $N = 50$  was drawn and a random noise term was added to each point. The random noise terms were generated from a Gaussian distribution with a mean of zero and a standard deviation of 0.5. Figure 5.8 displays the results of support vector regression and relevance vector regression applied to this data.



**Figure 5.8: Support vector regression (top) and relevance vector regression (bottom) applied to the simulated data.**

The red points in both figures present the observed data and the black curve is the function from which they were generated, as given in Equation 5.98. The blue curve in the top plot of Figure 5.8 shows the estimated regression line obtained using support vector regression. A total of 46 (out of 50) data points were used as support vectors to obtain this regression line. In comparison, the green line in the bottom plot of Figure 5.8 shows the regression line obtained using relevance vector regression. Amazingly, only 8 data points were identified as relevance vectors to obtain this regression line which appears to fit the true function possibly even better than the support vector regression line. Hence, even though both support vector regression and relevance vector regression do well to model the underlying function, the relevance vector regression resulted in a much sparser fit, requiring only 8 data points to fit the regression line in comparison to the support vector regression line which used 46 out of the 50 data points.



### 5.6.3 Relevance Vector Classification

The relevance vector machine concept can easily be extended to include classification problems, although relevance vector regression tends to be more popular (Tipping, 2001). We consider the binary classification problem with response variable  $y \in \{0,1\}$ . The model is now given by the form

$$f(x) = \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x)) \quad (5.99)$$

where  $\sigma(\cdot)$  denotes the logistic sigmoid function. In contrast to performing Bayesian logistic regression and simply assigning a Gaussian prior with a single precision parameter, we use the automatic relevance determination prior whereby a separate precision parameter is assigned to each corresponding weight parameter  $\theta_i$ .

We cannot analytically compute the necessary integral over  $\boldsymbol{\theta}$  as we did with the regression setting and so we use a Laplace approximation. For this, we need to initialize the hyperparameter  $\boldsymbol{\alpha}$  and construct a Gaussian distribution as an approximation to the posterior which then leads to an approximation for the marginal likelihood. We then maximize this marginal likelihood to obtain an updated estimate for  $\boldsymbol{\alpha}$  and this process is iterated until convergence.

We assume that the conditional distribution for the data can be modelled by a Bernoulli distribution, so that the log of the posterior distribution for this model is given by

$$\begin{aligned} \log p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\alpha}) &= \log(p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\alpha})) - \log p(\mathbf{y}|\boldsymbol{\alpha}) \\ &\propto \log p(\mathbf{y}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \\ &= \sum_{n=1}^N \left\{ y_n \log \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x_n)) + (1 - y_n) \log (1 - \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x_n))) \right\} - \frac{1}{2} \boldsymbol{\theta}^T \mathbf{A} \boldsymbol{\theta} + c \end{aligned} \quad (5.100)$$

where  $c$  is a constant and  $\mathbf{A} = \text{diag}(\alpha_i)$ . This can be maximized using iteratively reweighted least squares (IRLS). This is done by first fixing the  $\boldsymbol{\alpha}$  values and determining the mode of the distribution,  $\boldsymbol{\theta}^{\max}$ , and then using this mode to determine the Hessian matrix,  $\mathbf{H}$ , which can then be inverted to obtain the covariance matrix. Once the converged values for  $\boldsymbol{\theta}^{\max}$  and  $\mathbf{H}$  have been obtained, we use these to update the values of the hyperparameters  $\boldsymbol{\alpha}$  in an identical fashion to the regression case.

We use the Laplace approximation to determine a Gaussian distribution with mean equal to the mode of the posterior. We define  $q(\boldsymbol{\theta}) := \log p(\mathbf{y}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\boldsymbol{\alpha})$ . A Taylor series expansion of this function around an initial value  $\boldsymbol{\theta}^{(0)}$  gives

$$\log q(\boldsymbol{\theta}) \approx \log q(\boldsymbol{\theta}^{(0)}) - \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)})^T \mathbf{H}^{(0)} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)}) \quad (5.101)$$

where  $\mathbf{H}^{(0)}$  is the Hessian matrix defined by

$$\begin{aligned}
\mathbf{H}^{(0)} &= \frac{\partial}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \log q(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(0)}} \\
&= \frac{\partial}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \left\{ \sum_{n=1}^N \left\{ y_n \log \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x_n)) + (1 - y_n) \log (1 - \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x_n))) \right\} - \frac{1}{2} \boldsymbol{\theta}^T \mathbf{A} \boldsymbol{\theta} + c \right\} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(0)}} \\
&= \sum_{n=1}^N \sigma(\boldsymbol{\theta}^{(0)T} \boldsymbol{\phi}(x_n)) [1 - \sigma(\boldsymbol{\theta}^{(0)T} \boldsymbol{\phi}(x_n))] \cdot \boldsymbol{\phi}(x_n)^T \boldsymbol{\phi}(x_n) + \mathbf{A} \\
&= \boldsymbol{\Phi}^T \mathbf{B}^{(0)} \boldsymbol{\Phi} + \mathbf{A}.
\end{aligned} \tag{5.102}$$

Here,  $\mathbf{B}^{(0)}: (N+1) \times (N+1)$  is a diagonal matrix with diagonal elements given by

$$b_{nn} = \sigma(\boldsymbol{\theta}^{(0)T} \boldsymbol{\phi}(x_n)) [1 - \sigma(\boldsymbol{\theta}^{(0)T} \boldsymbol{\phi}(x_n))]. \tag{5.103}$$

We then update this estimate of  $\boldsymbol{\theta}^{(0)}$  by iteratively reweighted least squares. At the  $j$ th iteration of the algorithm, we perform the Newton update

$$\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - (\mathbf{H}^{(j)})^{-1} \cdot \frac{\partial}{\partial \boldsymbol{\theta}^{(j)}} \{\log q(\boldsymbol{\theta})\} \tag{5.104}$$

where

$$\frac{\partial}{\partial \boldsymbol{\theta}} \{\log q(\boldsymbol{\theta})\} = \boldsymbol{\Phi}^T (\mathbf{y} - \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x))) - \mathbf{A} \boldsymbol{\theta} \tag{5.105}$$

with  $\sigma(\boldsymbol{\theta}^T \mathbf{x}) = [\sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x_1)), \dots, \sigma(\boldsymbol{\theta}^T \boldsymbol{\phi}(x_N))]^T$ .

On convergence of the algorithm, this gradient will be equal to zero and so the final update will be given by

$$\boldsymbol{\theta}^{\max} \rightarrow \mathbf{A}^{-1} \boldsymbol{\Phi}^T (\mathbf{y} - \sigma((\boldsymbol{\theta}^{\max})^T \boldsymbol{\phi}(x))). \tag{5.106}$$

Using this converged estimate,  $\boldsymbol{\theta}^{\max}$ , we update our estimate of the Hessian matrix to be

$$\mathbf{H}^{\max} = \boldsymbol{\Phi}^T \mathbf{B}^{\max} \boldsymbol{\Phi} + \mathbf{A} \tag{5.107}$$

from which we obtain the covariance of the gaussian distribution by taking the inverse of this. Hence, it follows that

$$q(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\theta}^{\max}, (\mathbf{H}^{\max})^{-1}) \tag{5.108}$$

and since  $p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\alpha}) \propto q(\boldsymbol{\theta})$ , we have that the Laplace approximation to the posterior is given by this Gaussian distribution. This approximation was obtained for fixed values of the hyperparameters  $\boldsymbol{\alpha}$ . We can update their estimates using the same formula for the estimates given previously, that is, we update

$$\hat{\alpha}_i = \frac{\gamma_i}{(\theta_i^{\max})^2} \quad (5.109)$$

where  $\gamma_i = 1 - \alpha_i(\mathbf{H}_{ii}^{\max})^{-1}$ .

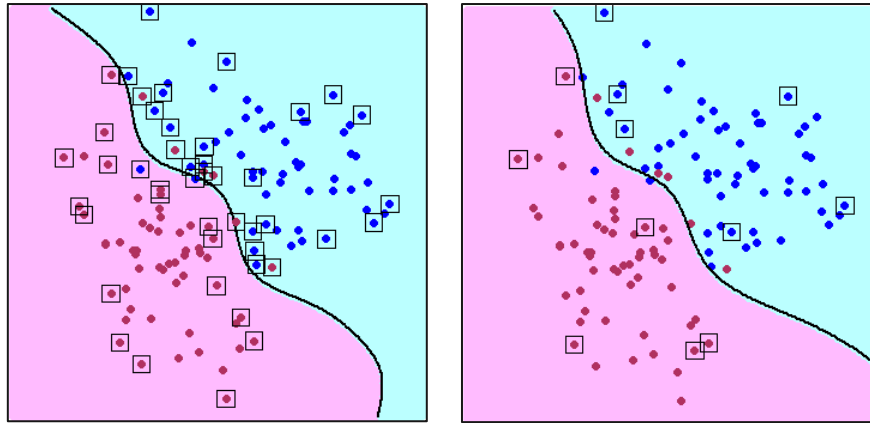
Using the new parameter updates, we return to the estimation of  $\theta^{\max}$  and repeat the entire process until any appropriate convergence criteria is met. Once converged, the process will yield a Gaussian approximation to the posterior as well as estimates for the hyperparameters.

We now compare the performance of the relevance vector classifier to the support vector machine. The data were simulated from two bivariate Gaussian distributions to create a blue and red class, with means and covariance matrices given respectively by

$$\boldsymbol{\mu}_{\text{blue}} = [1, 1]^T \quad \text{and} \quad \boldsymbol{\mu}_{\text{red}} = [-1, -1]^T \quad (5.110)$$

$$\boldsymbol{\Sigma}_{\text{blue}} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{\text{red}} = \begin{bmatrix} 1.5 & -0.8 \\ -0.8 & 2 \end{bmatrix}. \quad (5.111)$$

The resulting fit of the support vector machine and relevance vector classifier are shown in Figure 5.9.



**Figure 5.9: Support vector machine (left) and relevance vector classifier (right).**

The support vectors and relevance vectors identified respectively by each of the two methods have boxes drawn around them in. It is immediately clear to see the higher degree of sparsity in the fitted model obtained from the relevance vector classifier. In particular, the relevance vector classifier identified only 12 relevance vectors, corresponding to 12 basis functions, necessary in the model. In comparison, the support vector machine identified a total of 54 support vectors out of a possible 100 data points.

A further 200 data points were generated independently with 100 data points generated from each class. The SVM obtained a test error of 13.5% and the relevance vector classifier obtained a test error of only 9%. Hence, not only did the relevance vector classifier obtain a smaller test error, it

made use of only 12 relevance vectors to obtain this fit in comparison to the 54 support vectors for the SVM.

## **5.7 SUMMARY**

Variable selection techniques form a crucial part of any analysis since a reduction in the total number of variables in a model can improve the resulting fit of the model and decrease the computational time required. Reducing the number of variables also reduces the complexity of the model and it is naturally advantageous to rather work with a simpler model with fewer parameters if this reduced complexity does not significantly affect the overall performance of the model.

Of the variable selection techniques considered in this chapter, the Bayesian ridge regression with reversible jump MCMC resulted in the lowest test error for the data, however, all of the methods improved the overall test error when compared with the full model fit using a frequentist approach to linear regression. Since the training and test datasets were also not very large, the models were all comparably fast to implement.

The relevance vector machine was also considered as a model that rivals the support vector machine. In both examples provided, the relevance vector machine obtained a slightly improved fit and resulted in a significantly sparser model with much fewer relevance vectors required in comparison to the support vector machine.

## CHAPTER 6

### PROBABILISTIC GRAPHICAL MODELS

#### 6.1 GRAPH THEORY

Graphical models (GMs) are visual representations of the dependence and/or independence relations between the variables of a distribution. Roughly speaking, GMs fall broadly into two different categories; namely, those that are most useful for modelling and those that are best used for inference (Barber, 2011: 65).

The sum and product rule for probabilities form the backbone of essentially all calculations for probabilistic inference that are encountered. Probabilistic graphical models (PGMs) provide a convenient diagrammatic representation of probability distributions and their properties. Despite being a simple and graphical way to represent the structure of a model, they provide the user with a straightforward way of immediately identifying the properties of a probability distribution through inspection of the graph alone – the most useful being the conditional independencies.

Graphs are made up of a set of nodes, or vertices, which are joined through links, or edges. In the context of PGMs, the nodes represent random variables (or groups thereof) and the probabilistic dependencies and relationships between the variables are represented via links connecting the respective nodes. These edges can either be directed or undirected.

Bayesian networks and Markov models, as described next, are an example of directed graphical models where the links between nodes have an associated directionality denoting a specific conditional dependency between those random variables. The other major type of PGMs are undirected graphical models i.e. graphs containing links with no inherent direction. Naturally, directed graphs will prove to be more useful in scenarios where the key interest is the casual relationships between the variables, whereas undirected graphical models are typically more associated with problems of an inferential nature (Bishop, 2006: 360). This chapter will be most heavily focussed on the applications of graphical models in various contexts, with less attention given to the technicalities of the graph theory that constitute them.

#### 6.2 DIRECTED GRAPHICAL MODELS

##### 6.2.1 Bayesian Belief Networks

A Bayesian belief network, also known as a Bayes net (BN), is a graphical tool that is used to represent the independence assumptions made amongst the variables in a distribution, thereby introducing probabilistic structure into a model.

Representing the variables of a model by the set  $\{x_1, x_2, \dots, x_K\}$ , and the associated joint distribution function by  $p(x_1, x_2, \dots, x_K)$ , repeated application of the product rule results in the following equivalent expression of the joint distribution function in terms of a product of conditional distributions,

$$p(x_1, x_2, \dots, x_K) = p(x_K|x_1, \dots, x_{K-1}) \dots p(x_2|x_1)p(x_1). \quad (6.1)$$

This relationship can be expressed as a directed graph with  $K$  nodes for each of the variables, and each of the nodes having directed links pointing towards them from all nodes with a lower numbering. As an example of such a representation, we consider the simple case of  $K = 5$  variables, giving us,

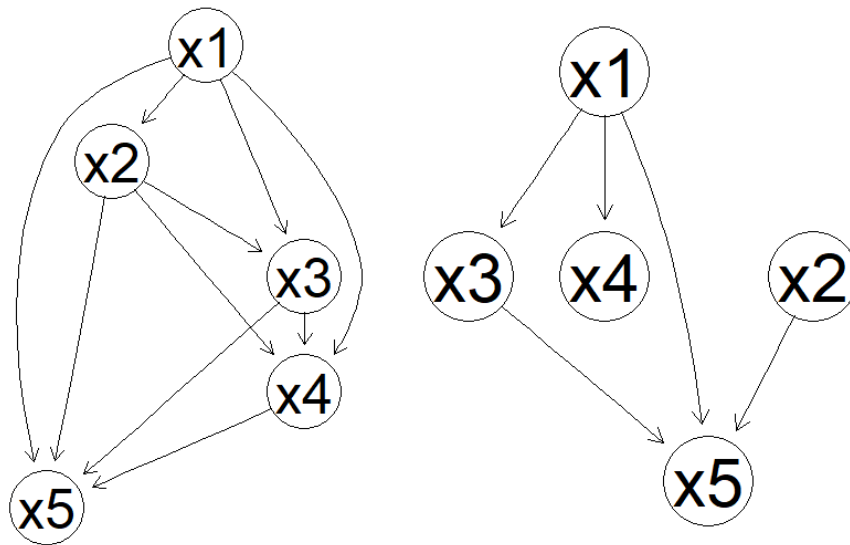
$$p(x_1, x_2, x_3, x_4, x_5) = p(x_5|x_1, x_2, x_3, x_4)p(x_4|x_3, x_2, x_1)p(x_3|x_2, x_1)p(x_2|x_1)p(x_1) \quad (6.2)$$

which is shown as a directed graph in the left-hand plot in Figure 6.1.

This is an example of a fully connected graph, since all nodes have a link between them. In practice, what is of greater interest is typically not the presence of links, but rather the absence of them that highlights interesting properties of the distributions. Suppose that there were certain conditional independencies present in the data so that the decomposition in Equation 6.2 could equivalently be expressed as

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_5|x_1, x_2, x_3)p(x_4|x_1)p(x_3|x_1)p(x_2)p(x_1). \quad (6.3)$$

This graphical model is displayed on the right in Figure 6.1.



**Figure 6.1: Fully connected, directed, acyclic graph (left) and the same graph but containing conditional independencies (right).**

Clearly, this graph is no longer fully connected, and we can immediately see how some of the edges from the left plot of Figure 6.1 have been removed due to this new decomposition. These removed edges indicate the conditional probabilities. It is important to remember that these two graphical models represent the same underlying joint distribution between the variables (Bishop, 2006: 362).

To fully define a belief network, we firstly define what is meant by a child and a parent node. If we have any two nodes, say, node  $a$  and node  $b$ , where there is a directed link leading from node  $a$  to node  $b$ , then node  $a$  is called the parent of node  $b$  and node  $b$  is the child of node  $a$ . By considering the right-hand graph in Figure 6.1, we can see that  $x_5$  is a child of  $x_2$ , and  $x_1$  is the parent of  $x_3$ ,  $x_4$  and  $x_5$ . This is just one such example of a child and parent node in this figure.

A Bayesian belief network is hence defined as a directed, acyclical, graph where the joint distribution of the random variables  $x$  can be expressed in the following factorized form:

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_i | \text{pa}(x_i)). \quad (6.4)$$

Here,  $\text{pa}(x_i)$  represents the set of parent nodes of the variable  $x_i$ . The  $i$ th node in the belief network graph corresponds to the factor  $p(x_i | \text{pa}(x_i))$ . This formulation is referred to as the chain rule for Bayesian networks (Koller & Friedman, 2009: 54). The added constraint of being acyclical means that if we consider any one of the nodes, we cannot find a route between the nodes such that we can return to our originally selected node. In other words, there are no loops, or cycles, in the graph. Clearly, both graphs in Figure 6.1 are acyclic.

Once the graph structure of the joint distribution has been defined, the relationships between the variables need to be quantified and this is done via the use of conditional probability tables (CPTs). Clearly this can become a huge task when there are many nodes, but one of the major benefits of Bayesian networks is that only nodes which are related in some probabilistic way are modelled which means that fewer conditional probabilities are required to define the joint distribution. This saves hugely on storage of probability values and makes complex models simpler, yielding a faster overall computation time.

To obtain the alternative decomposition in Equation 6.3, we assumed that there were some conditional dependencies present in the distribution. For example, the equality

$$p(x_4 | x_3, x_2, x_1) = p(x_4 | x_1) \quad (6.5)$$

means that variable  $x_4$  is conditionally independent of  $x_2$  and  $x_3$ . In other words, varying the values of  $x_2$  and  $x_3$  will not affect the value of  $x_4$  since this variable depends solely on the variable  $x_1$ . This essentially resulted in the directed edge  $x_2 \rightarrow x_4$  and the directed edge  $x_3 \rightarrow x_4$  in the left-hand graph in Figure 6.1 being removed to obtain the right-hand graph in Figure 6.1.

There exists a criterion known as *d*-separation (direct-dependent separation) that can be used to decide whether conditional independence is present in any nodes given a specific graph structure. We can define *d*-separation fairly simply by considering the nodes (or sets of nodes) **A**, **B** and **C**. The set of nodes **C** is said to *d*-separate the two other sets of nodes, **A** and **B**, if every path from a node in **A** to a node in **B** is blocked, given **C**. Hence, if **A** and **B** are *d*-separated by **C**, then **A** and **B** are conditionally independent given **C** (Bishop, 2006: 378).

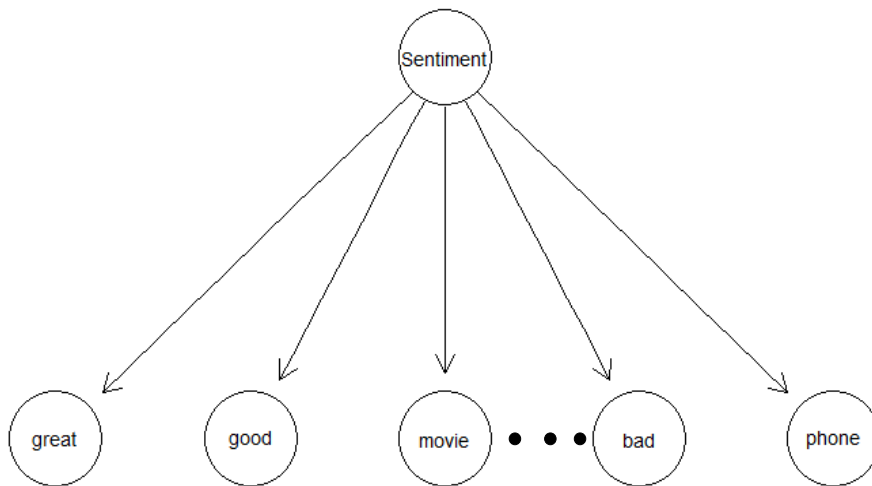
Another important concept when considering graphical models is the definition of the Markov blanket of a node. Given a node, **X**, the Markov blanket of this node is defined to be the set of nodes formed from the parents of **X**, the children of **X**, as well as any other node which shares the same child node as **X** (Murphy, 2012: 328). For example, the Markov blanket for node  $x_3$  in the right-hand graph of Figure 6.1 would be the set  $\{x_1, x_5, x_2\}$ . The important property of the Markov blanket is that once the Markov blanket of a node has been constructed, that node becomes *d*-separated from the rest of the network. In other words, a node is conditionally independent of all other nodes in the graph given the values of the nodes within its Markov blanket.

Any model that can be expressed in the form of a directed, acyclic graph defines a Bayesian Network. Clearly, we see that the Naïve Bayes modelling approach of Chapter 3 can be represented by a Bayesian network since the conditional independence assumption between the inputs results in the joint distribution function being expressed as

$$\begin{aligned} p(Y = k, \mathbf{x}) &= p(\mathbf{x}|Y = k)p(Y = k) \\ &= p(Y = k) \prod_{i=1}^p p(x_i|Y = k) \end{aligned} \quad (6.6)$$

which is exactly the factorized form of a BN. Hence, Figure 6.2 provides an example of how the naïve Bayes modelling approach applied to the sentiment dataset could be represented and computed as a BN. Of course, there would be a node for each of the inputs (words) but only 5 have been shown for illustrative purposes.



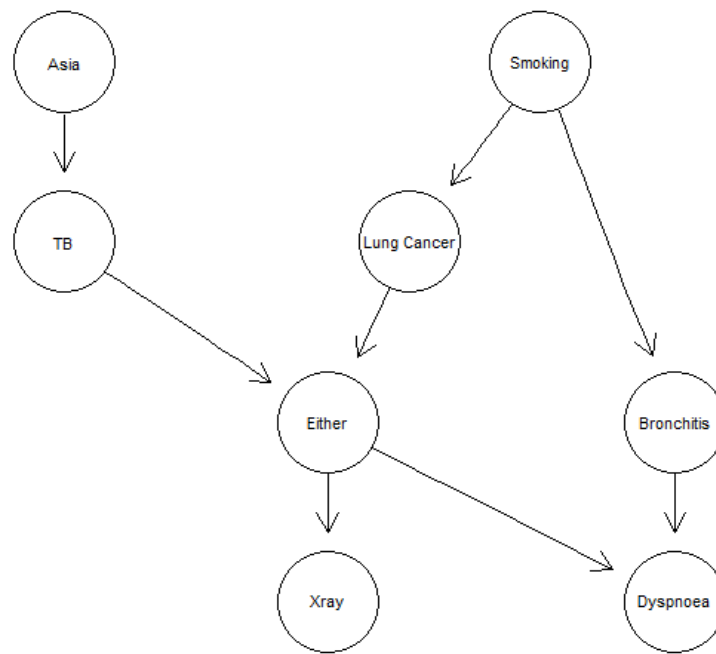


**Figure 6.2: Bayesian network representation of the Naïve Bayes model from Section 3.3.**

One of the most common applications of Bayesian networks is in medicine. We consider the following popular Asia dataset obtained from the bnlearn R package (Scutari, 2010) that is concerned with the relationships between lung diseases (tuberculosis (TB), lung cancer or bronchitis) and whether the person has visited Asia and/or whether the person is a smoker. The node ‘Either’ will refer to a person having either tuberculosis or lung cancer. We will assume that the graph structure is fixed and known and is the structure defined by Lauritzen & Spiegelhalter (1988) who created this synthetic dataset. Determining the structure of the graph is typically a very difficult task and often the structure is simply assumed in advance (Theodoridis, 2015: 837). Thus, the joint distribution function for the set of variables in the model is given as

$$\begin{aligned}
 & p(\text{Asia}, \text{TB}, \text{Smoking}, \text{Lung Cancer}, \text{Either}, \text{Bronchitis}, \text{Xray}, \text{Dyspnoea}) \\
 &= p(\text{Xray}|\text{Either})p(\text{Dyspnoea}|\text{Bronchitis}, \text{Either})p(\text{Either}|\text{TB}, \text{Lung Cancer}) \\
 &\quad \cdot p(\text{Bronchitis}|\text{Smoking})p(\text{Lung Cancer}|\text{Smoking}) \\
 &\quad \cdot p(\text{TB}|\text{Asia})p(\text{Smoking})p(\text{Asia})
 \end{aligned} \tag{6.7}$$

A BN of this dataset is provided in Figure 6.3.



**Figure 6.3: Bayesian network representation of the Asia dataset.**

BNs require two components to be fully defined. The first is the structure of the network, as given in Figure 6.3. The second component is the specification of the conditional probabilities. There are a total of 10 tables which together contain all the relevant conditional probabilities between the various nodes. For example, the conditional probability table for the parameters of node TB are given in Table 6.1. The full set of CPTs for this problem can be found in Appendix E.

**Table 6.1: Conditional probability table for the TB node.**

		Asia	
		No	Yes
TB	No	0.9915	0.9524
	Yes	0.0085	0.0476

The entries in this table are, for example, interpreted as

$$P(\text{TB} = \text{yes} | \text{Asia} = \text{yes}) = 0.0476 \quad (6.8)$$

The benefit of incorporating the conditional independencies into the model structure is that the total number of parameters to estimate is reduced from  $2^8 = 256$  to only 18. Each of these 18 parameters are those which are given in the CPTs, noting that, due to normalization, we do not need to set each entry of the CPT to be a parameter. For example, for the smoking node, we only

require the estimation of one parameter since we know that  $P(\text{Smoking} = \text{yes}) + P(\text{Smoking} = \text{no}) = 1$ . Hence, we only need to treat  $P(\text{Smoking} = \text{yes})$  as a parameter to estimate because we can use this value to determine  $P(\text{Smoking} = \text{no})$ , or vice versa.

Now that the whole BN has been defined, we can perform inference based on what we wish to determine. For example, a doctor may be interested in calculating the probability that a patient's X-ray will show lung disease based on various symptoms, or alternatively what the most likely cause of a patients symptoms are between the three different lung diseases. In general, Bayesian networks are typically used to determine the probability distribution for a variable (or set of variables) based on given values or evidence provided by other variables. Determining the probabilities of certain events based on new information is known as belief updating (Korb & Nicholson, 2010: 30).

It is, of course, possible to determine these probabilities by hand using the CPTs, but that is both tedious and time consuming. For example, suppose we were interested in determining the probability that a given person has dyspnoea given that they are a smoker. We can calculate this using the law of conditional probability as follows:

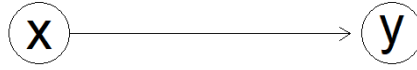
$$P(\text{Dyspnoea} = \text{yes} | \text{Smoking} = \text{yes}) = \frac{p(\text{Dyspnoea} = \text{yes}, \text{Smoking} = \text{yes})}{p(\text{Smoking} = \text{yes})} \quad (6.9)$$

Now, if we consider only the numerator,

$$\begin{aligned} & p(\text{Dyspnoea} = \text{yes}, \text{Smoking} = \text{yes}) \\ &= \sum_{A, TB, E, X, B, LC} p(\text{Dyspnoea} = \text{yes}, \text{Smoking} = \text{yes}, \text{Asia}, \text{TB}, \text{Either}, \text{Xray}, \text{Bronchitis}, \text{Lung Cancer}) \\ &= \sum_{A, TB, E, X, B, LC} p(\text{Xray} | \text{Either}) p(\text{Dyspnoea} = \text{yes} | \text{Bronchitis}, \text{Either}) p(\text{Either} | \text{TB}, \text{Lung Cancer}) \\ &\quad \cdot p(\text{Bronchitis} | \text{Smoking} = \text{yes}) p(\text{Lung Cancer} | \text{Smoking} = \text{yes}) p(\text{TB} | \text{Asia}) \\ &\quad \cdot p(\text{Smoking} = \text{yes}) p(\text{Asia}). \end{aligned} \quad (6.10)$$

This is clearly a time-consuming task to compute. Instead, efficient algorithms exist to perform these inference calculations. These algorithms are either exact or approximate inference algorithms. Since many machine learning tasks tend to involve a large number of observations and variables, approximate methods in the form of sampling from the distributions are often preferred for the sake of timeous executions.

However, for exact inference, the basic idea is to update the distributions of the variables based on observed values of the other variables. These updates are performed using Bayes' rule. To demonstrate this, consider the simple joint distribution provided in Figure 6.4 below.



**Figure 6.4: Joint distribution of two variables represented as a Bayesian network.**

Based on this graphical model, this joint distribution function can be expressed as

$$p(x, y) = p(y|x)p(x). \quad (6.11)$$

Suppose now that we observe the value of the variable  $y$  and we would like to determine the posterior distribution of the variable  $x$ , given this new information, or evidence, provided by  $y$ . The posterior distribution for  $x$  can be obtained using Bayes rule as well as the sum and product rules for probability, that is,

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \quad (6.12)$$

where

$$p(y) = \sum_{x'} p(y|x')p(x'). \quad (6.13)$$

Hence, Equation 6.12 then provides an expression of the joint distribution in two different forms, namely,

$$p(x, y) = p(y|x)p(x) = p(x|y)p(y) \quad (6.14)$$

This then means that the joint distribution can also be graphically represented in the same way as Figure 6.4 except with the arrow pointing in the opposite direction, that is, from  $y$  to  $x$ . Hence, as new evidence arrives, this evidence is passed forward through the chain of linked nodes to update their conditional values and then subsequently passed backwards to obtain the marginal distributions. Although not discussed here further, this trivial example is what forms the basis for the sum-and-product algorithm, which extends to the max-sum algorithm (Bishop, 2006: 402). These two algorithms are commonly used for performing exact inference in graphical models and full details of the algorithms can be found in Bishop (2006: 393–418) or Murphy (2012: 707–728).

### 6.2.2 Markov Chains

In most of the theory in previous chapters, we worked under the assumption that the data observations were all independently and identically distributed. Due to this, the likelihood function for the data could be expressed as the product of the individual marginal distributions, that is,

$$p(\mathcal{D}|\mathcal{M}) = p(x_1, x_2, \dots, x_T) = \prod_{t=1}^T p(x_t) \quad (6.15)$$

In many instances, specifically when we are working with data that arises due to some kind of sequential nature, this independence assumption becomes poor since there is now clearly underlying correlation present between the samples.

Hence, a Markov model, or Markov chain, as introduced in Chapter 2, can be expressed as

$$\begin{aligned} p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T) &= p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_2) \dots p(\mathbf{x}_T|\mathbf{x}_{T-1}) \\ &= p(\mathbf{x}_1) \prod_{t=2}^T p(\mathbf{x}_t|\mathbf{x}_{t-1}) \end{aligned} \quad (6.16)$$

assuming that we are working with discrete time steps (Murphy, 2012: 589).

Clearly the assumption that the immediate past  $\mathbf{x}_{t-1}$  captures all the necessary information about the future is a fairly strong one to make. It is possible to relax this assumption by using a higher-order Markov chain. However, if there are long-range correlations present in the variables, higher-order Markov chains become problematic since the number of parameters starts to increase hugely.

Markov chains, although simple in their nature, feature in many applications. One such application is their use in MCMC for sampling from posterior distributions as discussed in Chapter 2 and used throughout this thesis. Another interesting application of Markov chains is in the original formulation of Google's PageRank algorithm (Page & Brin, 1998). Surfing through the web was considered to be a stochastic process, and the PageRank algorithm is then regarded a model of a user's behaviour. Each website is modelled as a node and the links between the nodes represent the other pages to which the current page has a citation. A higher number of links leading into a node results in that web page being ranked as "more important" than others with fewer links. However, this is extended further. Not all links are given an equal weighting, meaning that a link leading into a node from a more important node will result in a higher vote of importance than one coming from a less important node. Clearly, this whole system can be modelled as a Markov chain. Naturally, this is a simplification of the overall algorithm, but provides a good illustration of how Markov models feature in everyday life.

### 6.2.3 Dynamic Bayesian Networks

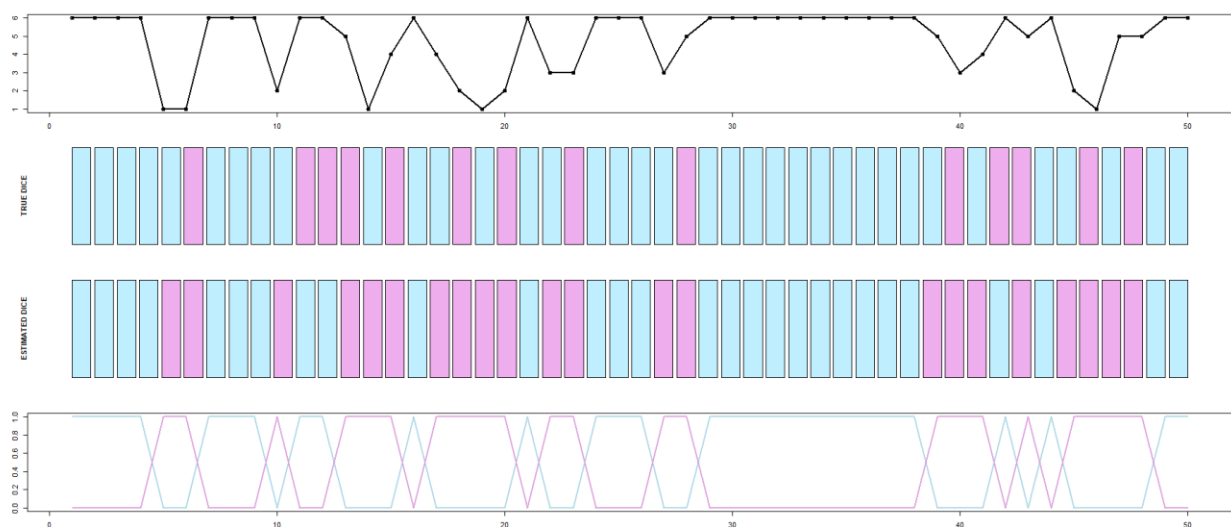
In Chapter 4, when we considered the linear regression problem, we had a fixed set of data and corresponding response variables to which we fit a regression model. We now extend this idea and consider the case where we wish to dynamically adapt a model that is changing over time. Such models are referred to as dynamic Bayesian networks (Bishop, 2006: 636). In these networks, it is not the actual graph structure which changes over time, rather that the graph

structure is representative of a dynamic system. We will consider two popular dynamic Bayesian networks; namely, hidden Markov models and linear dynamical systems.

### 6.2.3.1 Hidden Markov Models

A hidden Markov model (HMM) is a Markov chain that is defined on latent (i.e. ‘hidden’) variables. We begin the discussion of Hidden Markov models with a simple toy example to aid in the understanding of HMMs and the motivation for their use. Suppose that we have two dice: a pink die and a blue die. The pink die is unbiased and so the probability of rolling any one of the six numbers is  $1/6$ . On the other hand, suppose that the blue die was heavily biased, and the probability of rolling each of the six numbers is given by  $\{0.08, 0.08, 0.08, 0.08, 0.08, 0.6\}$ , respectively. We begin with one of the dice at random. If that die comes up with a six, that die is rolled again. If not, the dice are switched, and the process continues in that manner.

We consider the scenario where the colour of the dice being rolled cannot be observed, and all that we know is what the sequence of rolled values is. This situation can be modelled using a hidden Markov model where the colour of the die being rolled represents the two unknown, or hidden, states. Figure 6.5 summarizes the results of this simple experiment. A total of 50 rolls of the dice were observed, and this observed sequence of outputs is displayed in the top panel of Figure 6.5. The two subsequent plots display the true colour of dice that was thrown, and the estimated dice colour obtained from the HMM. It is clear to see that the HMM does quite well to distinguish between the two hidden states. In fact, the HMM was able to correctly predict the state of the system in 36 out of the 50 dice rolls.



**Figure 6.5: Hidden Markov model fit to the dice rolling problem.**

As mentioned, a hidden Markov model (HMM) is a Markov chain defined on hidden variables. We denote these hidden variables by  $z_t, t = 1, \dots, K$  where  $K$  is the total number of hidden states in the model (Murphy, 2012: 312). The Markov assumption is further applied to these latent variables

so that the probability distribution of  $\mathbf{z}_t$  is conditionally dependent on the previous state, that is, we consider the conditional distribution  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ . Hence, it is clear where the name ‘hidden Markov model’ is derived from.

Fitting an HMM involves (1) determining the model parameters, and (2) finding the best sequence of hidden states to explain the observed sequence of outcomes. The model parameters can be evaluated using maximum likelihood estimation for which the EM algorithm is used. In the E-step of the EM algorithm, the Baum-Welch algorithm (Baum, 1972) is further used to estimate the posterior distributions over the latent variables. The Viterbi algorithm (Viterbi, 1967) can then be used to determine what the most likely sequence of hidden states in the model is. Each of these will be discussed briefly in turn.

It is common to model the latent variables using a 1-of- $K$  coding scheme so that they correspond to a  $K$ -dimensional binary vector. All of the conditional probabilities  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$  can be represented in a transition matrix  $\mathbf{A}$ , where the  $ij$ th entry of  $\mathbf{A}$  at time step  $t$  is given by

$$A_{ij} = p(z_{tj} = 1 | z_{(t-1)i} = 1) \quad (6.17)$$

and these entries satisfy

$$0 \leq A_{ij} \leq 1 \quad \text{with} \quad \sum_j A_{ij} = 1. \quad (6.18)$$

This means that the conditional distributions can be written as

$$p(\mathbf{z}_t|\mathbf{z}_{t-1}) = \prod_{j=1}^K \prod_{i=1}^K A_{ij}^{z_{(t-1)i} z_{tj}}. \quad (6.19)$$

The first latent variable,  $\mathbf{z}_1$ , requires some form of initial distribution since it will not have any parent nodes. The marginal distribution of this variable is typically represented as

$$p(\mathbf{z}_1|\boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_{1k}} \quad (6.20)$$

where  $\pi_k = p(z_{1k} = 1)$  and  $\sum_k \pi_k = 1$ .

To define the full joint distribution of the observed and latent variables, we require the specification of the conditional distribution of the observed variables given then latent variables. This conditional probability is referred to as the emission probability and is given by

$$p(\mathbf{x}_t|\mathbf{z}_t, \boldsymbol{\phi}) = \prod_{k=1}^K p(x_t|\phi_k)^{z_{tk}} \quad (6.21)$$

where the dependence on the model parameters,  $\boldsymbol{\phi}$ , has been included for completeness.

Finally, the joint probability of the observed and latent variables can be expressed as

$$p(\mathbf{x}, \mathbf{z} | \Phi) = p(\mathbf{z}_1 | \boldsymbol{\pi}) \prod_{t=2}^T p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{A}) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{z}_t, \phi) \quad (6.22)$$

where  $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ ,  $\mathbf{z} = \{\mathbf{z}_1, \dots, \mathbf{z}_T\}$  and  $\Phi = \{\boldsymbol{\pi}, \mathbf{A}, \phi\}$  (Bishop, 2006: 610–612).

The likelihood function for an HMM is thus given by

$$p(\mathbf{x} | \Phi) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z} | \Phi) \quad (6.23)$$

which is obtained from Equation 6.19 where the hidden variables have been marginalized out of the joint distribution function. However, this function cannot be expressed as a sum of individual terms for each  $\mathbf{z}_t$  and a closed form solution does not exist for the direct maximization of this function. Thus, the EM algorithm is used to iteratively determine the parameter values which maximize this function.

It is often of interest to determine what the most probable sequence of observed states in the model is. In the dice rolling example given at the beginning of this section, the most probable sequence of states was returned by the algorithm and provided a sequence denoting which dice was most likely to have been rolled at each step of the experiment. Determining this most probable state sequence can be done using the Viterbi algorithm (Viterbi, 1967). This algorithm not only computes the most probable state sequence, but also provides the probability of being in that state. These probabilities are shown for each of the two dice in the bottom plot of Figure 6.5.

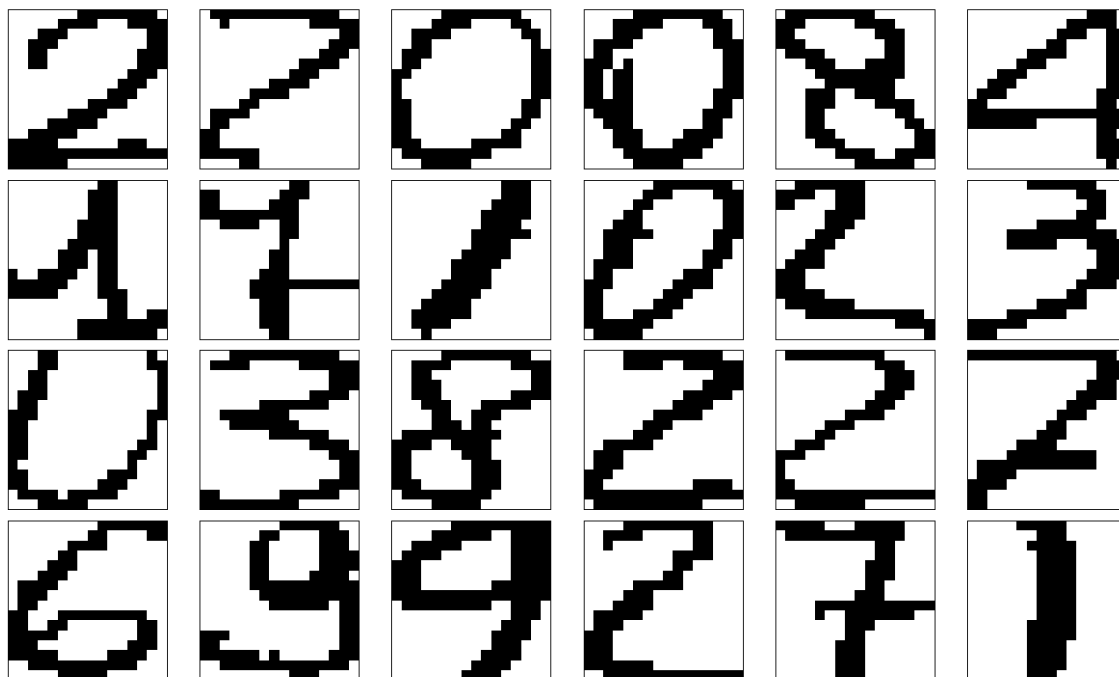
The basic idea of the algorithm is to recursively determine the most probable path by determining the probability of each hidden state emitting the observed value. The path corresponding to the highest probability can then be backtracked through to determine what the sequence of hidden states was. In other words, the algorithm determines the maximum out of all probabilities of different paths.

Hidden Markov models are often used in the social sciences, and common applications include speech recognition and speech emotion recognition (Nwe, Foo & De Silva, 2003), gene sequence classification (Mesa *et al.*, 2016) and on-line and off-line handwriting recognition (Dolfing, 1998). We will now consider the example of off-line handwriting recognition using a Hidden Markov Model. Off-line handwriting recognition refers to a written digit at a single time-point, whereas on-line handwriting recognition records the movements of the written digit in real time to construct a time series of the writing.

The Semeion Handwritten Digit dataset (Semeion Research Center of Sciences of Communication, 1994) consists of  $N = 1593$  instances of handwritten digits. Each of these



observations were stretched into a  $16 \times 16$  pixel grid and each pixel was converted into a binary 0/1 digit. An example of 24 random digits is given in Figure 6.6.



**Figure 6.6: 24 random digits of the Semeion Handwritten Digit dataset.**

A hidden Markov model was fit to this dataset where the 256 binary digits for each observation were used as the observed sequence, and the true value of the written digit was the hidden state. After testing using a range of starting values, the fitted model managed to determine a best sequence of states which was equal to the true digits approximately 74% of the time, i.e. had an error rate of roughly 26%. Considering how poor some of the written digits were, and the fact that even the human eye would struggle to decode them, this is not such a bad result. However, an HMM is by no means necessarily the best model to be fit to this dataset but was rather just illustrative of such an application of HMMs. Since this was a labelled dataset, supervised learning methods are likely to be more appropriate.

### **6.2.3.2 Linear Dynamical Systems**

Hidden Markov models assume that the hidden states are all discrete. We now consider another dynamic model where there are again hidden states, except now the hidden states are continuous. These models are referred to as state space models, and, in particular, we will firstly consider the case where all the conditional probability distributions in the model are Gaussian. This special case is referred to as a linear dynamical system (LDS). The transition and emission distributions can hence be written in the following traditional form (Bishop, 2006: 637)

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \mathbf{w}_t \quad (6.24)$$

$$\mathbf{x}_t = \mathbf{C}\mathbf{z}_t + \mathbf{v}_t \quad (6.25)$$

$$\mathbf{z}_1 = \boldsymbol{\mu}_0 \quad (6.26)$$

where each of the noise terms follow zero-mean Gaussian densities given by

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}) \quad (6.27)$$

$$\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \quad (6.28)$$

$$\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_0). \quad (6.29)$$

Hence, the transition and emission distribution functions can be expressed as

$$p(\mathbf{z}_t | \mathbf{z}_{t-1}) \sim \mathcal{N}(\mathbf{A}\mathbf{z}_{t-1}, \boldsymbol{\Gamma}) \quad (6.30)$$

$$p(\mathbf{x}_t | \mathbf{z}_t) \sim \mathcal{N}(\mathbf{C}\mathbf{z}_t, \boldsymbol{\Sigma}) \quad (6.31)$$

with the first latent variable having initial distribution

$$p(\mathbf{z}_1) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0). \quad (6.32)$$

These model parameters,  $\{\mathbf{A}, \boldsymbol{\Gamma}, \mathbf{C}, \boldsymbol{\Sigma}, \boldsymbol{\mu}_0, \mathbf{V}_0\}$  are determined using the Kalman filtering algorithm which takes the form of the EM algorithm (Bishop, 2006: 642–644). Exact computation in the algorithm is possible since all distributions in the model are Gaussian. The algorithm recursively makes predictions based on the current observations and then updates these predictions in a weighted fashion when a new observation arrives – the estimates that hold higher degree of certainty are given a higher weighting. Once these model parameters are obtained, we can fit a Kalman filter to the dataset to model the underlying function or to make predictions of what the next time point will most likely be.

Let the marginal distributions be given by

$$p(\mathbf{z}_t | \mathbf{x}_1, \dots, \mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{V}_t). \quad (6.33)$$

Firstly, the prediction step (Murphy, 2012: 641) is straightforward to derive and is given by

$$\begin{aligned} p(\mathbf{z}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) &= \int p(\mathbf{z}_t | \mathbf{z}_{t-1}) p(\mathbf{z}_{t-1} | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) d\mathbf{z}_{t-1} \\ &= \int \mathcal{N}(\mathbf{A}\mathbf{z}_{t-1}, \boldsymbol{\Gamma}) \mathcal{N}(\boldsymbol{\mu}_{t-1}, \mathbf{V}_{t-1}) d\mathbf{z}_{t-1} \\ &= \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}) \end{aligned} \quad (6.34)$$

where

$$\mathbf{P}_{t-1} = \mathbf{A}\mathbf{V}_{t-1}\mathbf{A}^T + \boldsymbol{\Gamma}. \quad (6.35)$$

Using this, we can use Bayes rule to determine

$$\begin{aligned} p(\mathbf{z}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t) &\propto p(\mathbf{x}_t | \mathbf{z}_t) p(\mathbf{z}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) \\ &= \mathcal{N}(\mathbf{C}\mathbf{z}_t, \boldsymbol{\Sigma}) \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}) \\ &= \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{V}_t). \end{aligned} \quad (6.36)$$

From Chapter 4, we immediately obtain

$$\boldsymbol{\mu}_t = \mathbf{V}_t(\boldsymbol{\Sigma}^{-1}\mathbf{C}^T\mathbf{x}_t + \mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1}) \quad (6.37)$$

$$\mathbf{V}_t^{-1} = \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C} + \mathbf{P}_{t-1}^{-1}. \quad (6.38)$$

Using matrix identities as well as the Woodbury identity, we can rewrite these equations into the following form:

$$\begin{aligned} \mathbf{V}_t &= (\mathbf{P}_{t-1}^{-1} + \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C})^{-1} \\ &= \mathbf{P}_{t-1} - \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma})^{-1}\mathbf{C}\mathbf{P}_{t-1} \\ &= (\mathbf{I} - \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma})^{-1}\mathbf{C})\mathbf{P}_{t-1} \\ &= (\mathbf{I} - \mathbf{K}_t\mathbf{C})\mathbf{P}_{t-1}. \end{aligned} \quad (6.39)$$

Also,

$$\begin{aligned} \boldsymbol{\mu}_t &= \mathbf{V}_t(\mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}_t + \mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1}) \\ &= (\mathbf{P}_{t-1}^{-1} + \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C})^{-1}(\mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}_t + \mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1}) \\ &= (\mathbf{P}_{t-1}^{-1} + \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}_t + (\mathbf{P}_{t-1}^{-1} + \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C})^{-1}\mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1} \\ &= \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma})^{-1}\mathbf{x}_t + (\mathbf{P}_{t-1}^{-1} + \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C})^{-1}\mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1} \\ &= \mathbf{K}_t\mathbf{x}_t + (\mathbf{P}_{t-1}^{-1} + \mathbf{C}^T\boldsymbol{\Sigma}^{-1}\mathbf{C})^{-1}\mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1} \\ &= \mathbf{K}_t\mathbf{x}_t + (\mathbf{P}_{t-1} - \mathbf{P}_{t-1}\mathbf{C}^T(\boldsymbol{\Sigma} + \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T)^{-1}\mathbf{C}^T\mathbf{P}_{t-1})\mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1} \\ &= \mathbf{K}_t\mathbf{x}_t + \mathbf{P}_{t-1}\mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1} - \mathbf{P}_{t-1}\mathbf{C}^T(\boldsymbol{\Sigma} + \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T)^{-1}\mathbf{C}^T\mathbf{P}_{t-1}\mathbf{P}_{t-1}^{-1}\mathbf{A}\boldsymbol{\mu}_{t-1} \\ &= \mathbf{K}_t\mathbf{x}_t + \mathbf{A}\boldsymbol{\mu}_{t-1} - \mathbf{P}_{t-1}\mathbf{C}^T(\boldsymbol{\Sigma} + \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T)^{-1}\mathbf{C}^T\mathbf{A}\boldsymbol{\mu}_{t-1} \\ &= \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{K}_t(\mathbf{x}_t - \mathbf{C}^T\mathbf{A}\boldsymbol{\mu}_{t-1}). \end{aligned} \quad (6.40)$$

Hence, we have that

$$\boldsymbol{\mu}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{K}_t(\mathbf{x}_t - \mathbf{C}\mathbf{A}\boldsymbol{\mu}_{t-1}) \quad (6.41)$$

$$\mathbf{V}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\mathbf{P}_{t-1} \quad (6.42)$$

where the matrix  $\mathbf{K}_t$  is known as the Kalman gain matrix, and is given by

$$\mathbf{K}_t = \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \boldsymbol{\Sigma})^{-1}. \quad (6.43)$$

Hence, starting with the initial distribution  $p(\mathbf{z}_1) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ , at time point  $t$ , given the new observation  $\mathbf{x}_t$  as well as the values of  $\boldsymbol{\mu}_{t-1}$  and  $\mathbf{V}_{t-1}$ , we are able to determine the marginal distribution for  $\mathbf{z}_t$ , which will follow a Gaussian distribution with mean  $\boldsymbol{\mu}_t$  and covariance  $\mathbf{V}_t$ . The set of equations defining the prediction step and the update steps, that is, Equations 6.34, 6.35, 6.41, 6.42 and 6.43 are combined referred to as the Kalman filtering equations (Bishop, 2006: 641).

One of the first applications of linear dynamical systems was to track the trajectory of a moving object. In fact, Kalman filtering was used in the navigational systems of the Apollo project in the 1960s. In particular, Kalman filters were used in the manned spacecraft missions to obtain estimates of the trajectories of the spacecraft on its journey to the moon and back (Grewal & Andrews, 2010). Since then, many variants and extensions of Kalman filters have been proposed, one of which will be discussed in the next section.

Kalman filters have been proven to be particularly useful in modelling and predicting time series data. We consider the simple univariate random walk model with noise given by

$$x_t = z_t + v_t \quad (6.44)$$

$$z_t = z_{t-1} + w_t \quad (6.45)$$

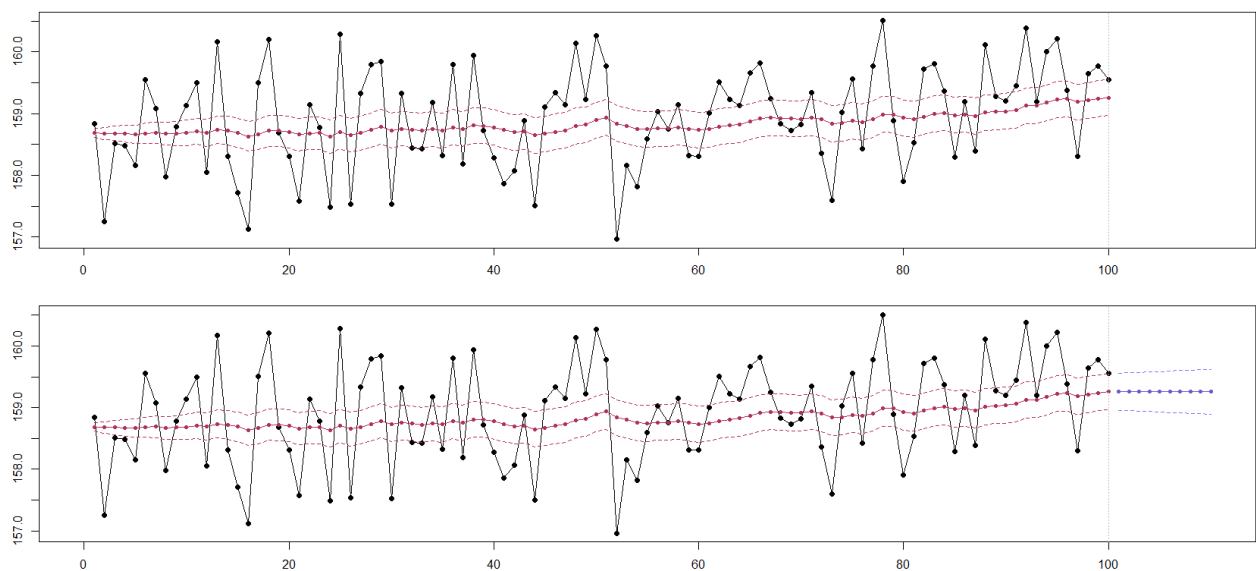
where

$$v_t \sim \mathcal{N}(0, \sigma) \quad \text{and} \quad w_t \sim \mathcal{N}(0, \gamma). \quad (6.46)$$

and the initial distribution for  $z_0$  is given by

$$z_0 \sim \mathcal{N}(\mu_0, \sigma_0). \quad (6.47)$$

This random walk model was constructed using values  $\sigma = 0.8$ ,  $\gamma = 0.1$ ,  $\mu_0 = 0$  and  $\sigma_0 = 100$ . The black line in Figure 6.7 represent the observed random walk model for the 100 time steps, and the red line are the resulting mean posterior points obtained from applying the Kalman filtering algorithm to the data. The dashed red lines show the 95% credible interval of these estimates. The bottom plot in Figure 6.7 also includes the 10-ahead forecasts in blue obtained from the Kalman filter as well as their 95% credible intervals.



**Figure 6.7: Output of the Kalman filtering algorithm applied to a simple random walk model.**

The Kalman filter does well to model the noisy time series data and is fast and easy to implement. The benefit of the Kalman filter is that it does not base estimates on a single measurement alone, but rather averages over measurements and uses the previous observation to guide the prediction of a future observation.

### 6.2.3.3 Particle Filters

Linear dynamical systems are those where both the conditional probability distributions in the data are Gaussian. We now consider the extended scenario where the emission probabilities possibly follow a non-Gaussian distribution, or, more importantly, are multimodal. Kalman filters are not applicable in these situations, hence, we introduce particle filtering. Particle filtering is also known as sequential Monte Carlo since it is a variant of Monte Carlo sampling. It is a recursive method whereby distributions are approximated by discrete random measures (Theodoridis, 2015: 854).

We suppose that we have observed data  $x_1, \dots, x_t$  and that we want to sample from the posterior distribution  $p(z_t | x_1, \dots, x_t)$ . We know that

$$\begin{aligned}
 E[p(z_t)] &= \int p(z_t) p(z_t | x_1, \dots, x_t) dz_t \\
 &= \int p(z_t) p(z_t | x_1, \dots, x_{t-1}, x_t) dz_t \\
 &= \frac{\int p(z_t) p(x_t | z_t) p(z_t | x_1, \dots, x_{t-1}) dz_t}{\int p(x_t | z_t) p(z_t | x_1, \dots, x_{t-1}) dz_t}, \quad \text{using Bayes' rule} \\
 &\approx \sum_{m=1}^M w_{tm} p(z_{tm})
 \end{aligned} \tag{6.48}$$

where the sampling weights  $w_{tm}$  are defined by

$$w_{tm} = \frac{p(x_t | z_{tm})}{\sum_{i=1}^M p(x_t | z_{ti})}. \tag{6.49}$$

In Equation 6.48 and 6.49, the  $z_{tm}$  are values sampled from the distribution  $p(z_t | x_1, \dots, x_{t-1})$ . We have now obtained a set of samples and a corresponding set of weights at time step  $t$ . Since we are working with sequential data, we assume now that we observe a new observation,  $x_{t+1}$ , and want to determine the new weights and samples for time point  $t + 1$ . It is easy to sample from  $p(z_{t+1} | x_1, \dots, x_t)$ , since, in a similar manner to the derivation of Equation 6.48, it follows that,

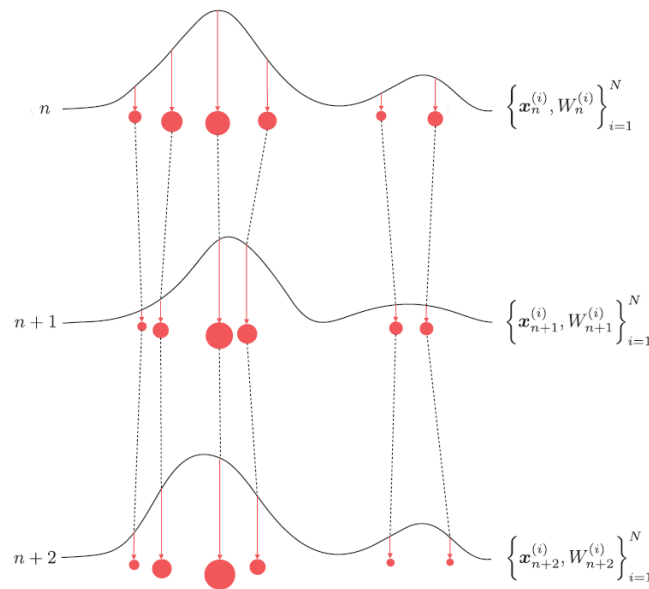
$$p(z_{t+1} | x_1, \dots, x_t) \approx \sum_{m=1}^M w_{tm} p(z_{t+1} | z_{tm}). \tag{6.50}$$

This is simply a mixture distribution from which sampling is straightforward.

Hence, beginning with initial values, the particle filtering algorithm evolves in two stages of iterations. Firstly, at a given time step  $t$ , we have  $M$  samples and weights from the posterior distribution  $p(\mathbf{z}_t | \mathbf{x}_1, \dots, \mathbf{x}_t)$ . Then, to determine the analogous representation of the posterior distribution for the next time point, we make use of the mixture distribution form of Equation 6.50 and sample from it accordingly, using the sampled observation  $x_{t+1}$  to determine the corresponding weights at time point  $t + 1$  (Bishop, 2006: 645–646), that is,

$$w_{t+1,m} \propto p(x_{t+1} | z_{t+1,m}). \quad (6.51)$$

A visual example of how a general particle filter works is provided in Figure 6.8 for 3 time steps and 6 particles.

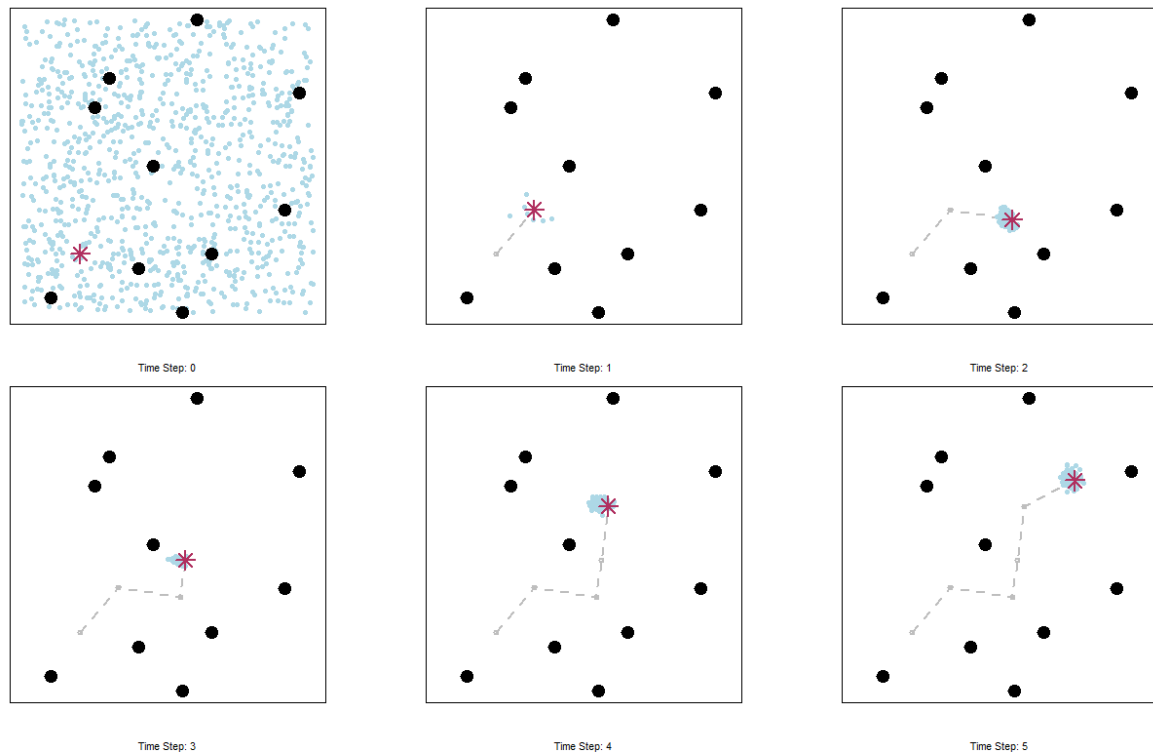


**Figure 6.8: Visual representation of a general particle filter.**

Source: Theodoridis, 2015: 857

As mentioned, Kalman filters, although powerful, are not applicable in situations where the data are multimodal or non-linear. Thrun *et al.*, (2001) proposed the use of particle filters in a process referred to as mobile robot localization. Particle filters have become particularly famous in this area and are forming the backbone to many autonomous motor vehicle programs. There are many variants of this algorithm, but we will focus here on the simple case of a mobile robot moving in a two-dimensional space with various obstacles. The aim is for the robot to determine its location in this space using a sensor or radar that is only able to measure how far away it is from the obstacles. This example was adapted from code obtained from GitHub that was constructed to demonstrate localization of a self-driving car (Kuman, 2017).

The process over 5 different time steps is shown in Figure 6.9. The true location of the robot in space is shown by the red star, the obstacles are the black circles, and the small blue circles display the particles at that time step of the algorithm. The gray dashed shows the historic movement of the robot over time.



**Figure 6.9: Robot localization using particle filters.**

At time point zero, the particles are distributed uniformly over the entire space. This uniform distribution over the space represents each of the possible locations that the robot could be in. At time step 1, the robot takes a step and then scans the space to determine the distance between itself and each of the obstacles. A small random noise term is added to this sensor since in real life it will not be 100% accurate. All the particles are shifted in a similar manner to the robot (in order to predict the next location of the robot) and their distances to each of the obstacles is calculated. Since those particles that have a similar location in space to the robot will have similar distances to each of the obstacles, these particles are given a higher weighting than the rest. The particles are then randomly sampled proportionate to this weighting, but since not all particles will appear in the final sample due to having an almost zero weighting, the particle space becomes smaller and more concentrated to the true region of the robot.

This example, although a simplification of the real world, demonstrates the power of particle filters and their applications in robotics and object localization. A more impressive use of particle filters was proposed by Smith & Cheeseman (1986) and is referred to as simultaneous localization and

mapping (SLAM). In this task, the obstacles themselves are now unknown, and the aim is to determine the robot's location in space whilst mapping out the space simultaneously. Hence, as the robot follows along some path in the space, it uses the measurements it obtains to construct a map representing the space in which it moves. Examples of other applications of particle filters in areas such as aircraft and car navigation, tracking or terrain mapping can be found in Gustafsson *et al.*, (2002).

## 6.3 UNDIRECTED GRAPHICAL MODELS

### 6.3.1 Markov Networks

Bayesian belief networks are an example of a directed, acyclical graph. We now consider undirected graphs, also known as Markov networks or Markov random fields (MRFs). Analogously to BNs, each node represents a random variable, except now the edges connecting the respective nodes are no longer directed and hence indicate no preference to the direction of dependence (Theodoridis, 2015: 763).

Markov networks will be informally introduced through the use of an example. A common application of Markov networks is in the study of image denoising. Consider Figure 6.10. The left-hand plot shows the original image which was constructed from a binary pixel matrix with entries  $\{-1, 1\}$  representing which pixels are black and which are white. The middle two plots show the original image that has been corrupted with noise. The top middle plot has been corrupted with approximately 10% noise such that each pixel had a 10% probability of switching colour at random. Similarly, the bottom middle plot has been corrupted with 20% noise in the same manner.

De-noising an image (i.e. restoring it to the original image) can be done using Markov random fields. The observed sequence of nodes are the pixels given in the noisy plot, and the unknown nodes are each of the true colours for the nodes. For each of these observed nodes, we can calculate its probability distribution based on the values of its neighbouring nodes (i.e. its Markov blanket) since a pixel in an image has a high probability of looking very similar to its surrounding pixels. In other words, pixels are highly correlated with their neighbours, especially when the level of noise is not too high, and this prior knowledge is what is captured in the Markov random field. This probability distribution over the pixels can then be used to denoise the image and restore it (as far as possible) to the original image. Details of this method will be provided shortly.



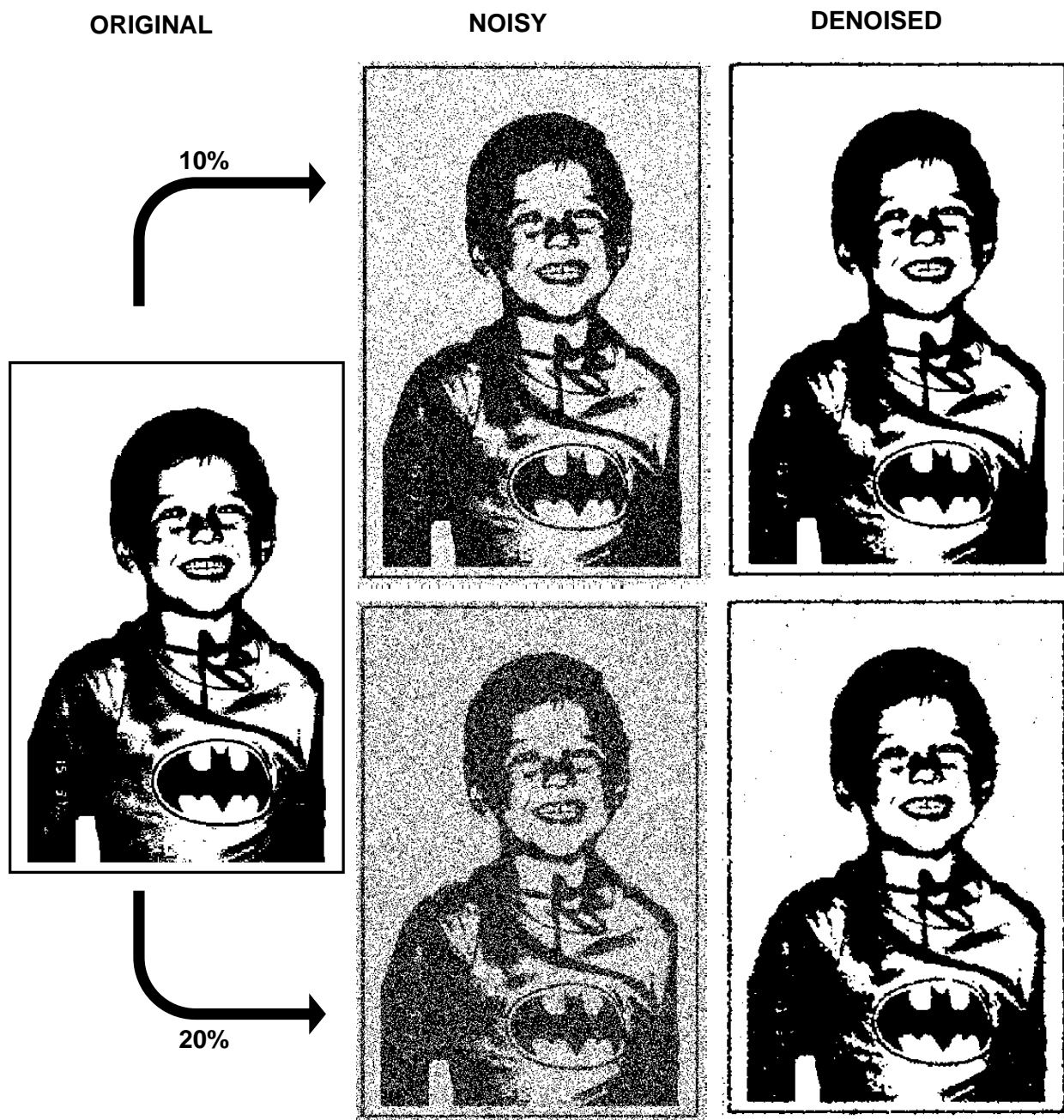
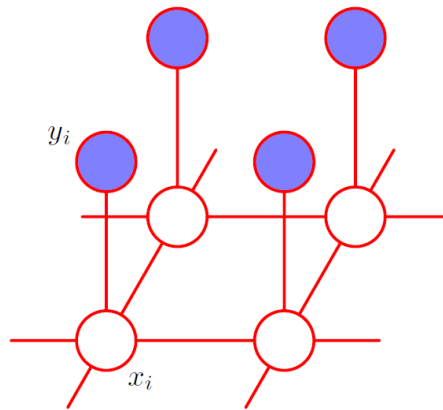


Figure 6.10: Markov random field applied to the image denoising problem.

A simple illustration of how this undirected graphical model looks is given in Figure 6.11. The nodes denoted by  $y_i$  represent the observed nodes in the noisy image, and the blue shaded nodes denoted by  $x_i$  represent the unknown, true value of each pixel.



**Figure 6.11: Visual representation of the undirected graphical model for the image denoising problem.**

Source: Bishop, 2006: 389

The right-hand plots of Figure 6.10 show the denoised images corresponding to the noisy images to the left of them. Clearly, when the noise level is lower (i.e. 10% as opposed to 20%), the model is better able to obtain the original image since the pixels will be more highly correlated with each other. However, this model was still able to perform very well even with 20% noise.

To formally define a Markov network, we firstly require the definition of potentials and cliques. A potential, denoted by  $\psi(x)$ , is any non-negative function of the variable  $x$ , that is,  $\psi(x) \geq 0$ . More familiarly, a probability distribution is simply a potential with the added constraint of normalization, that is,  $\sum_x \psi(x) = 1$  assuming discrete  $x$ . Potential functions are also sometimes referred to as factors (Murphy, 2012: 665).

A clique is defined as any fully connected set of nodes where all nodes are neighbours. Further, a maximal clique is a clique whereby no larger cliques exist containing it (Barber, 2011: 30). In other words, even though a subset of nodes in a graph may be fully connected and thereby define a clique, if there is any other fully connected subset of nodes that contain all those nodes as well as others, then that original set of nodes forms a clique, but will not be maximal.

Hence, a Markov Network is an undirected graph defined as the following product of potentials

$$p(x|\theta) = \frac{1}{Z} \prod_{c=1}^C \psi_c(x_c|\theta_c) \quad (6.52)$$

where  $c$  indexes over the  $C$  maximal cliques in the graph and  $Z$  is the normalizing constant of the distribution, also referred to as the partition function (Barber, 2011: 66). This normalizing constant is given by

$$Z = \sum_{\mathbf{x}} \prod_c \psi_c(\mathbf{x}_c | \boldsymbol{\theta}_c). \quad (6.53)$$

A major advantage of parameterizing the model in this way is that it allows for greater flexibility with regards to representing the interactions amongst different variables (Koller & Friedman, 2009: 106).

When working with directed graphs, we parameterized the model using conditional probability tables. We parameterize an undirected graphical model in a similar way, except now we specify the graph structure by means of the potentials. This, unfortunately, means that the parameters in an undirected graph are not always as intuitive to understand as those in a directed graph.

Fitting a Markov network involves determining the maximum likelihood estimates for the parameters, but typically no closed form solution for these exist and even using gradient-based optimizers is not always possible (Murphy, 2012: 678). A common alternative is to rather maximize the pseudo log-likelihood function, that is, maximize over the full set of conditionals, which is given by

$$\ell_{pl}(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^p \log p(x_{ip} | x_{i,-d}, \boldsymbol{\theta}) \quad (6.54)$$

where the " $-d$ " notation refers to all those except the  $d$ th. Of course, each Markov network will look different to the next based on the choice of potential functions so the methods of determining the parameters and fitting the model will vary slightly.

We return again to the particular case of the binary image denoising problem and the representation of the graphical model given in Figure 6.11. This graph is made up of two different types of cliques. The first are cliques of the form  $\{x_i, y_i\}$  and the second are ones of the form  $\{x_i, x_j\}$ , where  $i$  and  $j$  index over the total number of pixels in the image. Since potential functions are strictly positive functions, they are often represented equivalently using exponentials, so that each potential is written as

$$\psi_c(\mathbf{x}_c) = \exp\{-En(\mathbf{x}_c)\} \quad (6.55)$$

where  $En(\mathbf{x}_c)$  is the energy function associated with that clique. In this case, since we know that neighbouring pixels should be more highly correlated with each other than other pixels further away, we respectively assign the energy functions  $-\eta x_i y_i$  and  $-\beta x_i x_j$  to each of the two types of cliques, where  $\eta, \beta > 0$ . Hence, the full energy function for this model can be given as

$$En(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i \quad (6.56)$$

where term  $hx_i$  is a bias term added to each pixel to favour them towards a certain sign. This term can be regarded as the prior probability assigned to each node. If we set  $h = 0$ , we are assigning equal probability (i.e. no preference) to either pixel value for that node. Further, setting  $\beta = 0$  would result in the best solution simply being given by  $x_i = y_i$  since there would be no links between any neighbouring nodes and the original noisy image will simply be returned. Hence, we can consider the values of the constants  $\beta$  and  $\eta$  to be penalty parameters. The constant  $\beta$  controls the penalty assigned when there are differences in the values of neighbours and  $\eta$  controls the penalty associated with changing the value of the current pixel under consideration. Larger values of these constants will force the resulting reconstructed image to be smoother.

The joint distribution function for this problem can then be expressed as

$$\begin{aligned}
 p(\mathbf{x}, \mathbf{y} | \beta, \eta) &= \frac{1}{Z} \prod_{c=1}^C \psi_c(\mathbf{x}_c | \boldsymbol{\theta}_c) \\
 &= \frac{1}{Z} \exp\{-E n(\mathbf{x}, \mathbf{y})\} \\
 &= \frac{1}{Z} \exp\left\{-h \sum_i x_i + \beta \sum_{\{i,j\}} x_i x_j + \eta \sum_i x_i y_i\right\}. \quad (6.57)
 \end{aligned}$$

This particular form of model is an example of the Ising model (Bishop, 2006: 389). The values of  $y$  are then set fixed to the observed pixel values in the noisy image, and the aim is to determine the value of the image  $x$  which maximizes the conditional distribution given by  $p(\mathbf{x} | \mathbf{y})$ . This can be solved using an algorithm known as iterated conditional modes (Kittler & Föglein, 1984).

The iterated conditional modes algorithm starts at some initial position, typically by setting  $x_i = y_i$ , and then moves through each pixel and calculates the total energy for that node based on its neighbours for each of the two classes. These neighbours correspond to the eight nodes surrounding each node, that is, for node  $(i, j)$  in the pixel matrix, its neighbours are given by

$(i - 1, j - 1)$	$(i - 1, j)$	$(i - 1, j + 1)$
$(i, j - 1)$	$(i, j)$	$(i, j + 1)$
$(i + 1, j - 1)$	$(i + 1, j)$	$(i + 1, j + 1)$

However, this is of course not applicable to nodes on the edges of the matrix or nodes in the corners since these nodes will subsequently have fewer neighbouring nodes.

The node  $x_i$  will then take on the same class value as the class which returned the largest value of the energy function given by Equation 6.56. This process is repeated until some stopping criteria is reached. Note that this algorithm will not always converge to the global maximum. This

algorithm is what was used to denoise the image given in Figure 6.10. For this example, the values of  $\beta$  and  $\eta$  were both set equal to 1. The denoised images in Figure 6.10 were obtained after just 1 iteration of the algorithm, that is, each entry in the noisy matrix (i.e. each node of the model) was visited only once.

Other popular and interesting uses of Markov random fields include image segmentation (Daily, 1989) and detecting moving objects in videos using a stationary camera (Subudhi, Ghosh & Ghosh, 2015) .

### 6.3.2 Conditional Random Fields

A conditional random field (CRF) is a variation of a Markov random field where the interest now lies in the conditional distribution of a set of observed variables  $X$  and a set of target variables  $Y$  (Theodoridis, 2015: 768). The structure of a CRF is given by

$$p(y|x) = \frac{1}{Z_x} \prod_{c=1}^c \psi(y_c|x_c) \quad (6.58)$$

where

$$Z_x = \sum_y p(y|x). \quad (6.59)$$

Conditional random fields are sometimes referred to as discriminative random fields, since, in comparison to MRFs where the model was a generative one, the focus is now discriminatory in nature because the interest is in modelling the distribution of the target variables given a set of observed predictors. However, in the same way that logistic regression requires labelled training data and naïve Bayes does not, the disadvantage of CRFs over MRFs is that CRFs require labelled training data whereas MRFs do not (Murphy, 2012: 684). Training a CRF follows in an analogous manner to that of MRFs.

One common application of CRFs is in named entity recognition (NER). This involves extracting information from a dataset and classifying it into pre-defined categories. As an example, we consider the NER dataset (Kaggle: NER\_dataset, 2020) which consists of  $N = 47\,959$  sentences of varying length. Each of these sentences has been broken down by word and labelled according to its part of speech as well as a named entity recognition tag. For example, the following sentence:

*“In other violence, U.S. officials said one American soldier was killed while on patrol in Baghdad Sunday.”*

is displayed in the data matrix in the form as shown in Table 6.2. The interpretation of this table is as follows: the first column denotes which sentence each word belongs to – in this case,

sentence 28. The second column denotes the actual word in the sentence. The third and fourth columns provide the standard assigned part-of-speech and named entity recognition tags, respectively. For example, “IN” refers to a preposition and “NN” to a noun. For the tag column, “O” denotes no specific tag, whereas “B-gpe” and “B-tim” denote a geopolitical entity and a time indicator, respectively.

**Table 6.2: Example of the data matrix for sentence number 28 in the NER dataset.**

Sentence	Word	POS	Tag
28	In	IN	O
28	other	JJ	O
28	violence	NN	O
28	,	,	O
28	U.S	NNP	B-gpe
28	officials	NNS	O
28	said	VBD	O
28	one	CD	O
28	American	JJ	B-gpe
28	soldier	NN	O
28	was	VBD	O
28	killed	VBN	O
28	while	IN	O
28	on	IN	O
28	patrol	NN	O
28	in	IN	O
28	Baghdad	NNP	B-geo
28	Sunday	NNP	B-tim
28	.	.	O

The final column which provides the named entity recognition tag is the tag that is of interest. The conditional random field aims to predict this tag by using the surrounding tags to infer what the

most likely tag should be. There are a total of 9 unique tags that were assigned. The dataset was split into a training set (50%) and test set (50%) and the resulting CRF obtained an error rate of only 4.97%.

## 6.4 SUMMARY

Probabilistic graphical models provide a convenient and simple way to represent the joint distribution of a set of random variables. The edges between nodes represent conditional dependencies and a lack of an edge represents a conditional independency. Visual inspection of these models provides the user with an immediate means of determining where the conditional dependencies and independencies in the model are. Further, by expressing the joint distribution function in terms of these conditional independencies, the total number of parameters in the model can be hugely reduced resulting in fewer estimations required.

Many of the techniques and algorithms encountered in the topics of PGMs rely heavily on the use of Bayes' theorem and the nodes in the graphical models are treated in a Bayesian framework: they are variables, or sets of variables, that are assumed to follow some unknown distribution. Conditional probabilities and independencies are central to many of the methods discussed in this chapter. Further, inference in graphical models also takes the form of a Bayesian approach whereby the distributions of the nodes are updated to follow some posterior distribution when new data, or evidence, arrives. This distributional assumption over the nodes also allows the uncertainty inherent in the state of a node to be directly quantified.

Probabilistic graphical models as a topic on its own has a vast literature with many applications. This chapter considered just a few of the more common and interesting graphical models from a mostly practical point of view since the mathematical theory behind many of the approaches becomes complex very quickly. From the simple, yet powerful, applications provided in this chapter, it is clear to see the value and relevance of graphical models due to their application in many areas of everyday life.



## CHAPTER 7

### CONCLUSION

The focus of this thesis was to provide a comprehensive overview of some of the most popular Bayesian machine learning techniques and their common applications. Several different Bayesian approaches to machine learning were considered and, where applicable, compared in terms of mathematical complexity, execution time and accuracy to the corresponding frequentist methods.

In both the classification and regression chapters, the Bayesian approach was compared with the frequentist approach of the same method. In all examples provided in these two chapters, the Bayesian approach showed valuable advantages related to obtaining a full posterior distribution over the parameter space, allowing for the uncertainty in the estimates to be immediately apparent and incorporated into the analysis. Further, the Bayesian approaches were also shown to reduce the issue of providing extreme conclusions. This was especially useful in the case of the Naïve Bayes modelling approach with regards to the problem of zero training instances for a particular class. The Bayesian approach allowed for the class conditional probabilities to essentially be smoothed which resulted in a lower test error.

The simulation example provided in Chapter 4 highlighted the major properties of Bayesian modelling; particularly the influence that the prior and likelihood can have on the posterior distribution. Further, the Bayesian linear regression approach required very few training samples before it started to obtain accurate results and a resulting predictive distribution with a small variance. Bayesian model comparison is also especially useful when the training data is not very large. In comparison with frequentist methods of model selection such as cross-validation, Bayesian model comparison provides a means of comparing different model choices using the training data alone. However, it was also shown that the Bayesian results converge towards those obtained from a frequentist approach as the size of the available data becomes very large. In this case, there are not significant advantages in a fully Bayesian approach apart from a full posterior distribution over the model parameters.

The four different variable selection techniques discussed in Chapter 5 all provided an improved fit in comparison to the full model that was fit using classical multiple linear regression. These techniques all relied heavily on MCMC sampling to approximate the posterior distribution of the model parameters but were all still efficient to implement. The relevance vector machine was further proposed as a sparse Bayesian learning technique that provides an alternative to the classical support vector machine. Both methods make use of a kernelized form of the regression equation and for both the regression and classification problems presented, the relevance vector machine obtained a much sparser model requiring fewer basis functions with a comparative, if not improved, fit.



The final chapter of this thesis presented several different probabilistic graphical models with a focus more towards their practical use rather than their theoretical backgrounds. PGMs form a prominent part in the Bayesian machine learning literature and this is evident through the examples and applications of these techniques in everyday life. Various fields of scientific literature, ranging from the social sciences to the biological sciences, make regular use of PGMs. Hence, this is a rapidly expanding field as the demand for greater automation and advanced artificial intelligence increases.

Both Bayesian and frequentist approaches have their place in the machine learning world, and neither should exist for sole use over another. Every dataset is different, and even two datasets obtained from a similar field of study may not necessarily be best suited to the same modelling approach, or even the same model. Bayesian methods do often provide valuable advantages over their frequentist counterparts, but sometimes these advantages are minimal and not worth the added complexity or computational time required. However, as computing power continues to improve, the field of Bayesian machine learning will most likely continue to expand alongside it.

## REFERENCES

- Andrews, D.F. & Mallows, C.L. 1974. Scale Mixtures of Normal Distributions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(1):99–102.
- Barber, D. 2011. *Bayesian Reasoning and Machine Learning*. Cambridge University Press.
- Baum, L.E. 1972. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. *Inequalities*, 3(1):1–8.
- Beaumont, M.A., Zhang, W. & Balding, D.J. 2002. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.
- Bishop, C.M. 2006. *Pattern Recognition and Machine Learning*. New York: Springer-Verlag.
- Casella, G. 2001. Empirical Bayes Gibbs sampling. *Biostatistics*, 2(4):485–500.
- Daily, M.J. 1989. Color image segmentation using Markov random fields. *Proceedings CVPR '89: IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [Online]. 4-8 June, San Diego, CA, USA: IEEE. 304–312. Available: <https://ieeexplore.ieee.org/document/37865> [2020, October 02].
- Dolfing, J.G.A. 1998. A comparison of ligature and contextual models for hidden Markov model based on-line handwriting recognition. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* [Online]. 15 May, Seattle, WA, USA: IEEE. Available: <https://ieeexplore.ieee.org/document/675454> [2020, September 15].
- Dua, D. & Graff, C. 2019. *UCI Machine Learning Repository* [Online]. Irvine, CA: University of California, School of Information and Computer Science. Available: <http://archive.ics.uci.edu/ml> [2020, July 10].
- Efron, B. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26.
- Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., Ishwaran, H., Knight, K., Loubes, J.M., Massart, P., et al. 2004. Least angle regression. *Annals of Statistics*, 32(2):407–499.
- Feinerer, I. & Hornik, K. 2019. *tm: Text Mining Package* [Online]. R package version 0.7-7. Available: <https://cran.r-project.org/package=tm> [2020, July 15].
- Freguglia, V. & Garcia, N.L. 2020. *mrf2d: Markov random field image models in R* [Online]. Available: <https://cran.r-project.org/package=mrf2d> [2020, September 21].
- Gabry, J. & Mahr, T. 2020. *bayesplot: Plotting for Bayesian Models* [Online]. R package version 1.7.2. Available: <https://mc-stan.org/bayesplot> [2020, June 30].
- Garcia-Donato, G. & Forte, A. 2018. Bayesian Testing, Variable Selection and Model Averaging

- in Linear Models using R with BayesVarSel. *The R Journal* [Electronic], 10(1):155–174. Available: <https://journal.r-project.org/archive/2018/RJ-2018-021/index.html> [2020, July 20].
- Geman, S. & Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.
- George, E.I. & Foster, D.P. 2000. Calibration and empirical bayes variable selection. *Biometrika*, 87(4):731–747.
- George, E.I. & McCulloch, R.E. 1993. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889.
- Goodrich, B., Gabry, J., Ali, I. & Brilleman, S. 2020. *rstanarm: Bayesian applied regression modeling via Stan* [Online]. R package version 2.21.1. Available: <https://mc-stan.org/rstanarm> [2020, July 26].
- Gramacy, R.B., Moler, C. & Turlach, B.A. 2019. *monomvn: Estimation for MVN and Student-t Data with Monotone Missingness* [Online]. R package version 1.9-13. Available: <https://cran.r-project.org/package=monomvn> [2020, August 19].
- Green, P.J. 1995. Reversible jump Markov chain monte carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.
- Grewal, M.S. & Andrews, A.P. 2010. Applications of Kalman Filtering in Aerospace 1960 to the Present. *IEEE Control Systems*, 30(3):69–78.
- Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R. & Nordlund, P.J. 2002. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437.
- Hastie, T., Tibshirani, R. & Friedman, J. 2009. *Springer Series in Statistics The Elements of Statistical Learning - Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.
- Hastie, T., Tibshirani, R. & Friedman, J. 2019. *ElemStatLearn: Data Sets, Functions and Examples from the Book: The Elements of Statistical Learning, Data Mining, Inference, and Prediction* [Online]. R package version 2015.6.26.2. Available: <https://cran.r-project.org/package=ElemStatLearn> [2020, July 20].
- Hastings, W.K. 1970. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.
- Ishwaran, H. & Rao, J.S. 2005. Spike and slab variable selection: Frequentist and bayesian strategies. *Annals of Statistics*, 33(2):730–773.
- Jaakkola, T.S. & Jordan, M.I. 1997. A variational approach to Bayesian logistic regression models

- and their extensions. *Sixth International Workshop on Artificial Intelligence and Statistics*, 82. Springer: 1–12.
- James, G., Witten, D., Hastie, T. & Tibshirani, R. 2013. *An Introduction to Statistical Learning with Applications in R*. Springer (ed.): Heidelberg, Berlin.
- Kaggle. 2020. *Kaggle: NER\_dataset* [Online]. Available: <https://www.kaggle.com/namanj27/ner-dataset> [2020, September 28].
- Kass, R.E. & Raftery, A.E. 1995. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.
- Kittler, J. & Föglein, J. 1984. Contextual classification of multispectral pixel data. *Image and Vision Computing*, 2(1):13–29.
- Koller, D. & Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series)*. Cambridge, Massachusetts, London, England: The MIT Press.
- Korb, K.B. & Nicholson, A.E. 2010. *Bayesian artificial intelligence, second edition*. United States of America: CHAPMAN & HALL/CRC.
- Kotzias, D., Denil, M., De Freitas, N. & Smyth, P. 2015. From group to individual labels using deep features. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* [Online]. August, Sydney, NSW, Australia: Association for Computing Machinery. 597–606. Available: <https://doi.org/10.1145/2783258.2783380> [2020, September 10].
- Kuman, A. 2017. pygame-robotics. *GitHub* [Online]. Available: <https://github.com/ioarun/pygame-robotics/blob/master/particle-filter/particle-filter-2.py> [2020, August 26].
- Kuo, L. & Mallick, B. 1998. Variable selection for regression models. *Sankhyā: The Indian Journal of Statistics, Series B, (Series B)*:65–81.
- Lauritzen, S.L. & Spiegelhalter, D.J. 1988. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–224.
- Liang, F., Paulo, R., Molina, G., Clyde, M.A. & Berger, J.O. 2008. Mixtures of g priors for Bayesian variable selection. *Journal of the American Statistical Association*, 103(481):410–423.
- Lindley, D. V. 1957. A Statistical Paradox. *Biometrika*, 44(1/2)(187).
- MacKay, D. 2005. *Information Theory, Inference, and Learning Algorithms*. Version 7. ed. Cambridge University Press.

- Malsiner-Walli, G. & Wagner, H. 2016. Comparing Spike and Slab Priors for Bayesian Variable Selection. *Austrian Journal of Statistics*, 40(4).
- Mesa, A., Basterrech, S., Guerberoff, G. & Alvarez-Valin, F. 2016. Hidden Markov models for gene sequence classification: Classifying the VSG gene in the *Trypanosoma brucei* genome. *Pattern Analysis and Applications*, 19(3):793–805.
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.
- Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. & Leisch, F. 2019. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien* [Online]. R package version 1.7-3. Available: <https://cran.r-project.org/package=e1071> [2020, July 21].
- Mitchell, T.J. & Beauchamp, J.J. 1988. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032.
- Murphy, K.P. 2012. *Machine learning: a probabilistic perspective (adaptive computation and machine learning series)*. MIT press.
- Neal, R. 1996. Bayesian Learning for Neural Networks. *Lecture notes in statistics*, 1(118).
- Nielsen, J.K., Christensen, M.G., Cemgil, A.T. & Jensen, S.H. 2014. Bayesian model comparison with the g-prior. *IEEE Transactions on Signal Processing*, 62(1):225–238.
- Nwe, T.L., Foo, S.W. & De Silva, L.C. 2003. Speech emotion recognition using hidden Markov models. *Speech Communication*, 41(4):603–623.
- Page, L. & Brin, S. 1998. The anatomy of a large-scale hypertextual Web search engine. *Proceedings of the Seventh World Wide Web Conference*, 30:107–117.
- Park, T. & Casella, G. 2008. The Bayesian Lasso. *Journal of the American Statistical Association*, 103(482):681–686.
- Petris, G. 2010. An R Package for Dynamic Linear Models. *Journal of Statistical Software* [Electronic], 36(12):1–16. Available: <http://www.jstatsoft.org/v36/i12/> [2020, August 20].
- Rasmussen, C.E. & Williams, C.K.I. 2018. *Gaussian Processes for Machine Learning*. The MIT Press.
- Rizzo, M.L. 2007. *Statistical Computing with R*. London, UK: Chapman & Hall/CRC.
- Rubin, D.B. 1984. Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician. *The Annals of Statistics*, 12(4):1151–1172.

- Scott, S.L. 2020. *BoomSpikeSlab: MCMC for Spike and Slab Regression* [Online]. R package version 1.2.3. Available: <https://cran.r-project.org/package=BoomSpikeSlab> [2020, August 16].
- Scutari, M. 2010. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software* [Electronic]. 35(3):1–22. Available: <http://www.jstatsoft.org/v35/i03/> [2020, September 16].
- Semeion Research Center of Sciences of Communication. 1994. *via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3-5, 25030 Mairano (Brescia), Italy* [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/semeion+handwritten+digit> [2020, September 15].
- Sherman, J. & Morrison, W.J. 1950. Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics* [Electronic], 21(1):124–127.
- Smith, R.C. & Cheeseman, P. 1986. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68.
- Subudhi, B.N., Ghosh, S. & Ghosh, A. 2015. Application of Gibbs–Markov random field and Hopfield-type neural networks for detecting moving objects from video sequences captured by static camera. *Soft Computing*, 19(10):2769–2781.
- Tavaré, S., Balding, D.J., Griffiths, R.C. & Donnelly, P. 1997. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518.
- Theodoridis, S. 2015. *Machine Learning: A Bayesian and Optimization Perspective*. Elsevier Ltd.
- Thrun, S., Fox, D., Burgard, W. & Dellaert, F. 2001. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141.
- Tipping, M.E. 2001. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1(3):211–244.
- de Valpine, P., Turek, D., Paciorek, C.J., Anderson-Bergman, C., Temple Land, D. & Bodik., R. 2017. Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, (26):403–413.
- Visser, I. & Speekenbrink, M. 2010. depmixS4: An R Package for Hidden Markov Models. *Journal of Statistical Software* [Electronic], 36(7):1–21. Available: <http://www.jstatsoft.org/v36/i07/> [2020, August 22].
- Viterbi, A.J. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.
- Wijffels, J. & Okazaki, N. 2018. *Conditional Random Fields for Labelling Sequential Data in*

- Natural Language Processing based on CRFsuite: a fast implementation of Conditional Random Fields (CRFs)* [Online]. Available: <https://github.com/bnosac/crfsuite> [2020, September 15].
- Xu, S., Li, Y. & Wang, Z. 2017. Bayesian multinomial naïve bayes classifier to text classification. *Lecture Notes in Electrical Engineering*. Singapore: Springer Verlag. 347–352.
- Zabaras, N. 2017. *Caterpillar Regression Example: Conjugate Priors, Conditional & Marginal Posteriors, Predictive Distribution, Variable Selection* [Online]. Notre Dame, IN, USA: University of Notre Dame. Available: <https://www.dropbox.com/s/pk2caw144nvzjcd/Lec12-CaterpillarRegressionExample.pdf?dl=0> [2020, August 19].
- Zellner, A. & Siow, A. 1980. Posterior odds ratios for selected regression hypotheses. *Trabajos de Estadística Y de Investigación Operativa*, 31(1):585–603.

## APPENDIX A

### CHAPTER 2 CODE

```

#-----#
# Generate the data from  $N(6, 1/4)$ 
#-----#

set.seed(1212)
tau <- 4
mu0 <- 0
tau0 <- 5
y <- rnorm(20, mean=6, sd=1/sqrt(tau))

# true values of the posterior
(muN <- (tau0*mu0+tau*sum(y))/(tau0+length(y)*tau))

## [1] 5.66341

(tauN <- tau0+length(y)*tau)

## [1] 85

#-----#
# Metropolis-Hastings algorithm
#-----#

theta <- 1
thetaVec <- NULL
for(i in 1:10000) {
  # generate a sample point
  theta.star <- rnorm(1, theta, 1)

  # acceptance ratio
  # use logarithms for stability
  alpha <- (sum(dnorm(y, theta.star, 1/sqrt(tau), log=T))+
            dnorm(theta.star, mu0, 1/sqrt(tau0), log=T))-
            (sum(dnorm(y, theta, 1/sqrt(tau), log=T))+
            dnorm(theta, mu0, 1/sqrt(tau0), log=T))

  # Accept or reject the sample point
  if(log(runif(1)) < alpha) {
    theta <- theta.star
  }

  # update the vector of accepted values
  thetaVec <- c(thetaVec, theta)
}

#-----#
# Trace plot and histogram with true density function
#-----#

par(mfrow=c(1,2))
plot(thetaVec, type='l', xlab="Iteration", ylab="Estimated value", col="maroon")

```



```

n")
hist(thetaVec[1001:10000], freq = F, breaks=20, main="", xlab="", col="lightblue")
x <- seq(from=5, to=6, length=1000)
post <- dnorm(x, mean=muN, sd=1/sqrt(tauN))
lines(x, post, type = 'l', lwd=2, col="maroon")

#-----#
# Compare the posterior values of the mean and precision
#-----#
mean(thetaVec[1001:10000])

## [1] 5.663974

1/sd(thetaVec[1001:10000])^2

## [1] 86.82422

#-----#
# GIBBS SAMPLING
#-----#

suppressMessages(library(mvtnorm))
suppressMessages(library(grDevices))

# Initialize the values
mu1 <- 1
mu2 <- -1
rho <- 0.8
sigma1 <- 1
sigma2 <- 1
cond.s1 <- sqrt((1-rho^2))*sigma1
cond.s2 <- sqrt((1-rho^2))*sigma2

# Starting values
x1 <- mu1
x2 <- mu2

#-----#
# Gibbs Sampler
#-----#

for(i in 1:10000) {
  cond.mu1 <- mu1+rho*sigma1/sigma2*(x2[i]-mu2)
  x1[i+1] <- rnorm(1, mean= cond.mu1, sd = cond.s1)
  cond.mu2 <- mu2+rho*sigma2/sigma1*(x1[i+1]-mu1)
  x2[i+1] <- rnorm(1, mean = cond.mu2, sd = cond.s2)
}

#-----#
# Trace Plots
#-----#

par(mfrow=c(2,1), mar=c(4,4,2,2))

```

```

plot(x1, type='l', col="purple", xlab="Iterations")
plot(x2, type='l', col="seagreen", xlab="Iterations")

# Obtain the means and covariance obtained from the Gibbs sampler
X <- cbind(x1, x2)[1001:10000,]
apply(X, 2, mean)

##           x1           x2
## 1.0185606 -0.9728463

cov(X)

##           x1           x2
## x1 0.9934652 0.7903589
## x2 0.7903589 0.9868477

# Plot and compare to true distribution
par(mfrow=c(1,1))
x <- seq(from=-3, to=3, length=1000)
xgrid <- expand.grid(x,x)
s12 <- rho*sigma1*sigma2

#-----#
# Simulated points from the Gibbs sampler (after burn-in of 1000)
#-----#
plot(X, col="plum2", xlim=c(-3,3),ylim=c(-3,3), xlab="", ylab="")

# True density function
filled.contour(x=x, y=x, z=matrix(dmvnorm(xgrid, mean=c(mu1,mu2),
                                         sigma = matrix(c(1,s12,s12,1),2,2, byrow=T))), nco
l=1000),
              col=cm.colors(n=27), xlim=c(-3,3), ylim=c(-3,3))

suppressMessages(library(nimble))

#-----#
# Generate the noisy data
#-----#

N <- 100
p <- 5
set.seed(1212)
X <- cbind(1, matrix(rnorm(N*p), nrow = N, ncol = p))
theta <- c(2,0,-2,0,4,-6)
y <- rnorm(N, X%*%theta, sd = 1)

#-----#
# Change code from Bugs to be used in R
#-----#

ModelCode <- nimbleCode({

  # non-informative prior for all parameters
  sigma ~ dunif(0, 20)

```

```

for(i in 1:numVars) {
  theta[i] ~ dnorm(0, sd = 100)
}

# generate the data from the model
for(i in 1:N) {
  pred.y[i] <- inprod(X[i, 1:numVars], theta[1:numVars])
  y[i] ~ dnorm(pred.y[i], sd = sigma)
}
})

#-----#
# Define model constants, inits and data
#-----#

# function requires lists
Constants <- list(N = 100, numVars = 6)
Initial <- list(sigma = 1, theta = rnorm(Constants$numVars))
Data <- list(y = y, X = X)

# construct and configure the nimble model
lmMod <- nimbleModel(code = ModelCode,
                     constants = Constants,
                     inits = Initial,
                     data = Data)

## defining model...
## building model...
## setting data and initial values...

## running calculate on model (any error reports that follow may simply reflect
## missing values in model variables) ...
## checking model sizes and dimensions...
## model building finished.

# Configure the RJMCMC
Config <- configureMCMC(lmMod)

## ===== Monitors =====
## thin = 1: sigma, theta
## ===== Samplers =====
## RW sampler (1)
##   - sigma
## conjugate sampler (6)
##   - theta[] (6 elements)

configureRJ(Config,
            targetNodes = 'theta',
            priorProb = 0.5,
            control = list(mean = 0, scale = .2))

# Build the RJMCMC the be run

```

```

mcmcRJ <- buildMCMC(Config)
compMod <- compileNimble(lmMod)

## compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compilation details.
## compilation finished.

CmcmcRJ <- compileNimble(mcmcRJ, project = lmMod)

## compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compilation details.
## compilation finished.

# Run the RJMCMC
set.seed(100)
samples <- runMCMC(CmcmcRJ, niter = 10000, nburnin = 1000)

## running chain 1...

## |-----|-----|-----|-----|
## |-----|-----|-----|-----|

#-----#
# TRACE PLOTS
#-----#

par(mfrow = c(3, 2))
plot(samples[, 'theta[1]'], type='l', main = "Intercept", col="maroon", ylab="", xlab="Iteration")
plot(samples[, 'theta[2]'], type='l', main = "Theta 1", col="maroon", ylab="", xlab="Iteration")
plot(samples[, 'theta[3]'], type='l', main = "Theta 2", col="maroon", ylab="", xlab="Iteration")
plot(samples[, 'theta[4]'], type='l', main = "Theta 3", col="maroon", ylab="", xlab="Iteration")
plot(samples[, 'theta[5]'], type='l', main = "Theta 4", col="maroon", ylab="", xlab="Iteration")
plot(samples[, 'theta[6]'], type='l', main = "Theta 5", col="maroon", ylab="", xlab="Iteration")

#-----#
# ACB rejection sampling algorithm
#-----#
ABC <- function(eps=1) {
  # Generate the bernoulli dataset
  y <- sample(c(0,1), size=50, replace=T, prob = c(0.3,0.7))
  yS <- sum(y)

  # True values of the posterior
  alpha <- sum(y)+1
  beta <- length(y)-sum(y)+1

  # Iterate the algorithm 10 000 times
  eps <- eps
  keep <- NULL
  for(i in 1:50000) {

```

```
# generate a parameter value from the prior
a <- runif(1)

# generate samples from the Likelihood using parameter value
LikeSamples <- rbinom(50, size=1, prob=a)
LikeS <- sum(LikeSamples)

# Compute the difference
distance <- abs(yS-LikeS)
if(distance < eps) {
  keep <- c(keep, a)
}
}

# Plot the estimated posterior vs the true posterior
x <- seq(from=0,to=1, length=1000)
post <- dbeta(x, alpha,beta)
plot(density(keep), ylim=c(0, max(post)), xlim=c(0,1), lwd=2, col="slateblue",
     main=paste0("Epsilon = ", eps))
lines(x, post, type='l', lwd=2, col="maroon")
}

set.seed(823)
par(mfrow=c(1,3))
for(e in c(30,5,1)) {
  ABC(eps = e)
}
```

## APPENDIX B

### CHAPTER 3 CODE

```

#-----#
# SIGMOID FUNCTION
#-----#
t <- seq(from=-8, to=8, by=0.1)
sigmoid <- function(t) {
  1/(1+exp(-t))
}

# Plot the sigmoid function
plot(t, sigmoid(t), type='l', ylab=expression(paste(sigma, "(t)")),
      main="Sigmoid Function", lwd=2)
abline(v=0, lty=2, col="pink")
abline(h=0.5, lty=2, col="pink")

#-----#
# Laplace - Comparative example
#-----#

#-----#
# Beta
#-----#

LaplaceBeta <- function(alpha=2, beta=2) {

  # Generate sequence of x values
  x <- seq(0,1,length=1000)

  # Write function as expression to differentiate
  fx <- expression((x^(alpha-1))*((1-x)^(beta-1)))
  fxlog <- expression(log(x^(alpha-1)*(1-x)^(beta-1)))

  # First and second derivatives
  df <- function(x) {
    eval(D(fx, name="x"))
  }
  df2 <- function(x) {
    eval(D(D(fxlog, name="x"), name="x"))
  }

  # Determine the mean and variance for the approximation
  theta.star <- uniroot(df, interval=c(0.001, 0.999))$root
  H <- -df2(x=theta.star)

  # Laplace Approximation function
  ApproxFunction <- function(x) {
    sqrt(H/(2*pi))*exp(-1/2*H*(x-theta.star)^2)
  }

  # Plotting
  yApprox <- ApproxFunction(x)

```

```

yFunc <- eval(fx)/beta(alpha,beta)
plot(x, yFunc, type='l', xlim=c(0,1), ylim=c(0,max(yFunc)+1), ylab="",xlab=
"",lwd=2,
      main=paste0("Beta (", alpha,"",beta, ")"), col="maroon")
points(x,yApprox, type='l', xlim=c(0,1),ylim=c(0,max(yFunc)+1),lwd=2,ylab="
", xlab="",col="slateblue1")
legend('topright', legend = c("True Function","Approximation"), lty=1, col=
c("maroon","slateblue1"),
      lwd=2)
}

#-----#
par(mfrow=c(1,2), pty='s')
LaplaceBeta(alpha=10, beta=12)
#-----#

#-----#
# Mixture Model
#-----#

LaplaceMixture <- function(mix=0.5) {

  # Generate sequence of x values
  x <- seq(-4,16,length=10000)

  # FUNCTION 1: Normal(0,1)
  # FUNCTION 2: Normal(8,4)

  # Write function as expression to differentiate
  func <- function(x) {
    mix*exp(-1/2*x^2)+(1-mix)*exp(-1/2*4*(x-8)^2)
  }
  fx <- expression(mix*exp(-1/2*x^2)+(1-mix)*exp(-1/2*4*(x-8)^2))
  fxlog <- expression(log(mix*exp(-1/2*x^2)+(1-mix)*exp(-1/2*4*(x-8)^2)))

  # First and second derivatives
  df <- function(x) {
    eval(D(fx, name="x"))
  }
  df2 <- function(x) {
    eval(D(D(fxlog, name="x"), name="x"))
  }

  # Determine the mean and variance for the approximation
  # The max will be one of the two means so we check which it is
  theta.star <- ifelse(func(0)>=func(8), 0,8)
  H <- -df2(x=theta.star)

  # Laplace Approximation function
  ApproxFunction <- function(x) {
    sqrt(H/(2*pi))*exp(-1/2*H*(x-theta.star)^2)
  }
}

```

```

# Plotting
yApprox <- ApproxFunction(x)
yFunc <- mix*dnorm(x)+(1-mix)*dnorm(x, mean=8, sd=2)
plot(x, yFunc, type='l', xlim=c(-4,16), ylim=c(0,max(c(yFunc, yApprox))),
     ylab="",xlab="",lwd=2, main="Gaussian Mixture Model", col="maroon")
points(x,yApprox, type='l', xlim=c(-4,16),ylim=c(0,max(c(yFunc,yApprox))),
       lwd=2,ylab="", xlab="", col="slateblue1")
legend('topright', legend = c("True Function","Approximation"), lty=1, col=
c("maroon","slateblue1"),
      lwd=2)
}

LaplaceMixture(mix=0.6)

#-----#
# Simulated Data Example
#-----#

suppressMessages(library(MASS))
suppressMessages(library(grDevices))

#-----#
# Generate the data from two Gaussians
#-----#

N <- 50
mu1 <- c(2,2)
mu2 <- c(-2,-2)
set.seed(1)
xclass1 <- mvrnorm(N, mu1, Sigma = matrix(c(1.5,-1,-1,1.5),2,2))
xclass2 <- mvrnorm(N, mu2, Sigma = matrix(c(2,-1,-1,2),2,2))

# Full dataset
Data <- cbind(rbind(xclass1, xclass2),c(rep(1,N), rep(0,N)))
X <- Data[,1:2]
y <- Data[,3]

#-----#
# Plot the data
#-----#

plot(xclass1, col="maroon", pch=20, xlim=c(-6,6), ylim=c(-6,6), xlab="", ylab
="")
points(xclass2, col="blue", pch=20, xlim=c(-6,6), ylim=c(-6,6))
abline(v=0, h=0, lty=2, col="gray")

#-----#
# Laplace Approximation Function
#-----#

logposterior <- function(theta) {
  px <- 1/(1+exp(-X*theta))
  logposterior <- sum(y*log(px)+(1-y)*log(1-px))-
    1/2*t(theta-mu0)%solve(V0)%theta-mu0)

```



```

    return(logposterior)
}

# Initialize the parameter vector
theta <- c(0,0)
# Set the prior distribution
mu0 <- c(0,0)
V0 <- diag(rep(100,2))

#-----#
# Perform the Laplace approximation
#-----#

optim.out <- optim(par=theta, fn=logposterior, control = list(fnscale=-1),
                  hessian = T)
(theta.max <- optim.out$par)

## [1] 2.703039 3.256300

Hessian <- optim.out$hessian
(CovarMat <- -solve(Hessian))

##           [,1]      [,2]
## [1,] 13.1230945  0.1797754
## [2,]  0.1797754 12.9644915

#-----#
# Visualize the posterior
#-----#

x.points <- y.points <- seq(-8,8,length.out=100)
grid <- as.matrix(expand.grid(x.points, y.points))

# Contours using theta.max (i.e. a point estimate)
z <- matrix(data=1/(1+exp(-grid%%as.matrix(theta.max))), nrow=100, ncol=100)
filled.contour(x.points, y.points,z, xlim=c(-5,5),ylim=c(-5,5),
               plot.axes = points(Data[,1:2], col=c(rep("maroon",50), rep("blue",50)), pch=16),
               col = cm.colors(n=20))

#-----#
# Contours using MC averaging
#-----#

zmc <- matrix(0, nrow=100, ncol=100)
S <- 50
for(i in 1:S) {
  set.seed(i)
  mcsample <- mvrnorm(n=1, mu=theta.max, Sigma=CovarMat)
  mcmat <- matrix(data=1/(1+exp(-grid%%as.matrix(mcsample))), nrow=100, ncol=100)
  zmc <- zmc + mcmat
}

#-----#

```

```

# Decision boundary with MC averaging
#-----#

filled.contour(x.points, y.points, z=zmc/S, xlim=c(-5,5),ylim=c(-5,5),
               plot.axes = points(Data[,1:2], col=c(rep("maroon",50), rep("blue",50)), pch=16),
               col = cm.colors(n=20))

#-----#
# FREQUENTIST APPROACH #
#-----#

library(ElemStatLearn)
data("SAheart")
head(SAheart)

##   sbp tobacco  ldl adiposity famhist typea obesity alcohol age chd
## 1 160   12.00 5.73   23.11 Present   49   25.30   97.20  52   1
## 2 144    0.01 4.41   28.61 Absent    55   28.87    2.06  63   1
## 3 118    0.08 3.48   32.28 Present   52   29.14    3.81  46   0
## 4 170    7.50 6.41   38.03 Present   51   31.99   24.26  58   1
## 5 134   13.60 3.50   27.78 Present   60   25.99   57.34  49   1
## 6 132    6.20 6.47   36.21 Present   62   30.77   14.14  45   0

str(SAheart)

## 'data.frame':   462 obs. of  10 variables:
## $ sbp      : int  160 144 118 170 134 132 142 114 114 132 ...
## $ tobacco  : num  12 0.01 0.08 7.5 13.6 6.2 4.05 4.08 0 0 ...
## $ ldl      : num  5.73 4.41 3.48 6.41 3.5 6.47 3.38 4.59 3.83 5.8 ...
## $ adiposity: num  23.1 28.6 32.3 38 27.8 ...
## $ famhist  : Factor w/ 2 levels "Absent","Present": 2 1 2 2 2 2 1 2 2 2 .
## ..
## $ typea    : int  49 55 52 51 60 62 59 62 49 69 ...
## $ obesity  : num  25.3 28.9 29.1 32 26 ...
## $ alcohol  : num  97.2 2.06 3.81 24.26 57.34 ...
## $ age      : int  52 63 46 58 49 45 38 58 29 53 ...
## $ chd      : int  1 1 0 1 1 0 0 1 0 1 ...

#-----#
# Training and Test data
#-----#

set.seed(1212)
train <- sample(1:nrow(SAheart), size = 0.7*nrow(SAheart), replace = F)
test  <- (1:nrow(SAheart))[-train]

#-----#
# Frequentist Logistic Regression Approach
#-----#

fit <- glm(chd ~ sbp+tobacco+ldl+famhist+obesity+alcohol+age, data=SAheart,
           family = binomial, subset = train)
summary(fit)

```

```
##
## Call:
## glm(formula = chd ~ sbp + tobacco + ldl + famhist + obesity +
##       alcohol + age, family = binomial, data = SAheart, subset = train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.7224  -0.8581  -0.5025   0.9666   2.3191
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.751353   1.123184  -3.340 0.000838 ***
## sbp           0.002748   0.006701   0.410 0.681717
## tobacco      0.083548   0.031813   2.626 0.008633 **
## ldl          0.153695   0.069589   2.209 0.027202 *
## famhistPresent 0.808761   0.265770   3.043 0.002342 **
## obesity      -0.015263   0.035052  -0.435 0.663238
## alcohol       0.001140   0.005085   0.224 0.822619
## age          0.037171   0.011801   3.150 0.001634 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 415.66  on 322  degrees of freedom
## Residual deviance: 346.91  on 315  degrees of freedom
## AIC: 362.91
##
## Number of Fisher Scoring iterations: 4

# Predictions and test error
pred <- predict(fit, newdata = SAheart[test,], type="response")
pred.class <- ifelse(pred>0.5, 1,0)
mean(pred.class!=SAheart[test,"chd"])

## [1] 0.2733813

#-----#
# Bayesian Logistic Regression
#-----#

suppressMessages(library(rstanarm))
suppressMessages(library(ggplot2))
suppressMessages(library(bayesplot))

# set up the model
SAheart$chd <- factor(SAheart$chd)
x <- model.matrix(chd ~ sbp+tobacco+ldl+famhist+obesity+alcohol+age-1, data=SAheart, subset=train)
y <- SAheart$chd

#-----#
# set the prior and estimate posterior
#-----#
```

```

prior <- normal(0,1000)
posterior <- stan_glm(chd ~ sbp+tobacco+ldl+famhist+obesity+alcohol+age, data
= SAheart,
                    family = binomial(link = "logit"),
                    prior = prior, prior_intercept = prior, QR=TRUE,
                    seed = 24, subset=train, iter=10000)

#-----#
# Plot posterior distribution: intervals
#-----#

pplot <- plot(posterior, plotfun="intervals", prob_outer = 1, prob=1)
pplot + geom_vline(xintercept = 0)

# Coefficient point estimates (median)
round(posterior$coefficients,6)

##      (Intercept)          sbp      tobacco          ldl famhistPresent
##      -3.794023      0.002826      0.087303      0.158741      0.825870
##      obesity      alcohol          age
##      -0.017091      0.001004      0.037824

# posterior predictions and test error for bayesian approach
postpred <- posterior_predict(posterior, newdata = SAheart[test,], draws = 10
000)
postpredclass <- ifelse(apply(postpred, 2, mean)>0.5,1,0)
mean(postpredclass!=SAheart[test, "chd"])

## [1] 0.2589928

#-----#
# BAYESIAN LOGISTIC REGRESSION: LAPLACE APPROXIMATION
#-----#

# Set the response as a factor
SAheart$famhist <- as.numeric(SAheart$famhist)-1

x <- as.matrix(cbind(1, SAheart[train,-c(4,6,10)]))
y <- SAheart[train,"chd"]
ynum <- as.numeric(y)-1
N <- nrow(x)

#-----#
# Specify the prior distribution
#-----#

mu0 <- c(0,0,0,0,0,0,0,0)
V0 <- diag(rep(1000,8))

#-----#
# Function to evaluate the Log of the posterior distribution
#-----#

logposterior <- function(theta) {
  px <- 1/(1+exp(-x%*%theta))

```

```

logposterior <- sum(ynum*log(px)+(1-ynum)*log(1-px))-
  1/2*t(theta-mu0)%*%solve(V0)%*%(theta-mu0)
return(logposterior)
}

#-----#
# Initialize theta and perform the Laplace approximation
#-----#

theta <- c(0,0,0,0,0,0,0,0)
set.seed(1212)
optim.out <- optim(par=theta, fn=logposterior, control = list(fnscale=-1), he
ssian = T)
theta.max <- optim.out$par
round(theta.max, 6)

## [1] -0.633798 -0.000410 0.098407 0.239395 0.567010 -0.111715 -0.002938
## [8] 0.025173

Hessian <- optim.out$hessian
(CovarMat <- -solve(Hessian))

##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.2625545582 -4.461877e-03 5.670180e-03 4.211485e-03 -0.0191164739
## [2,] -0.0044618765 4.671198e-05 -9.577351e-06 -2.503100e-05 0.0001246796
## [3,] 0.0056701798 -9.577351e-06 1.058064e-03 5.367396e-05 0.0001813766
## [4,] 0.0042114848 -2.503100e-05 5.367396e-05 5.227885e-03 -0.0009947905
## [5,] -0.0191164739 1.246796e-04 1.813766e-04 -9.947905e-04 0.0714481268
## [6,] -0.0256674797 -3.783991e-05 -1.092224e-04 -8.403242e-04 -0.0001609575
## [7,] 0.0001096019 -5.667851e-06 -2.745173e-05 2.344233e-05 -0.0001335220
## [8,] 0.0001058698 -1.985574e-05 -1.239181e-04 -1.091885e-04 -0.0005014086
##           [,6]      [,7]      [,8]
## [1,] -2.566748e-02 1.096019e-04 1.058698e-04
## [2,] -3.783991e-05 -5.667851e-06 -1.985574e-05
## [3,] -1.092224e-04 -2.745173e-05 -1.239181e-04
## [4,] -8.403242e-04 2.344233e-05 -1.091885e-04
## [5,] -1.609575e-04 -1.335220e-04 -5.014086e-04
## [6,] 1.516241e-03 3.244599e-06 -7.850659e-05
## [7,] 3.244599e-06 2.784894e-05 4.094922e-06
## [8,] -7.850659e-05 4.094922e-06 1.304840e-04

#-----#
# Add a density plot over the MCMC sample histograms
#-----#

par(mfrow=c(3,3))
sims <- posterior$stanfit@sim # obtain the full set of MCMC samples
ParNames <- names(posterior$coefficients)
for(i in 1:8) {
  mcmcSamples <- sims$samples[[1]][[i]] # Obtain all mcmc samples

  # Set up plotting limits
  xlims <- c(min(min(mcmcSamples), theta.max[i]-3*sqrt(CovarMat[i,i])),
            max(max(mcmcSamples), theta.max[i]+3*sqrt(CovarMat[i,i])))
  MCxx <- seq(from=xlims[1], to=xlims[2], length=1000)

```

```

yy <- dnorm(MCxx, mean = theta.max[i], sd=sqrt(CovarMat[i,i])) # Laplace ap
proximation density
maxy <- max(yy, max(density(mcmcSamples)$y))

# histogram of mcmc samples and marginal Laplace
hist(mcmcSamples, main=ParNames[i], breaks=20, xlab="", col="slategray1", f
req = F,
      xlim=xlims, ylim=c(0, maxy))
lines(MCxx, yy, lwd=2, col="maroon")
}

#-----#

xtest <- as.matrix(cbind(1, SAheart[test,-c(4,6,10)]))
ytest <- SAheart[test,"chd"]

#-----#
# Predictions
#-----#

library(MASS)
pclass1 <- function(xnew, nsize) {
  mcsample <- matrix(data=mvrnorm(nsize, mu=theta.max, Sigma=CovarMat),
                    nrow=nsize, ncol=8)
  probs <- mean(1/(1+exp(-mcsample%*%xnew)))
  class <- ifelse(probs > 0.5,1,0)
  return(class)
}

class.pred <- NULL
for(j in 1:nrow(xtest)) {
  class.pred[j] <- pclass1(xtest[j,], 10000)
}

#-----#
# test error using Laplace approximation
#-----#

mean(class.pred!=ytest)

## [1] 0.2661871

#-----#
# NAIVE BAYES
#-----#

# Libraries needed
suppressMessages(library(tm))
suppressMessages(library(wordcloud))
suppressMessages(library(e1071))
suppressMessages(library(colorRamps))
suppressMessages(library(naivebayes))

#-----#

```

```

# Load in the data into one dataframe from files
amazon <- read.delim(file="amazon_cells_labelled.txt", header=F, sep="\t", quote="",
                    col.names = c("words", "sentiment"))
imdb <- read.delim(file="imdb_labelled.txt", header=F, sep="\t", quote="",
                  col.names = c("words", "sentiment"))
yelp <- read.delim(file="yelp_labelled.txt", header=F, sep="\t", quote="",
                  col.names = c("words", "sentiment"))

SentimentData <- rbind(amazon, imdb, yelp)

# Sentiments must be factors, sentences must be characters
SentimentData$sentiment <- factor(SentimentData$sentiment, levels = c(0,1),
                                labels = c("Negative", "Positive"))
SentimentData$words <- as.character(SentimentData$words)
summary(SentimentData)

##      words      sentiment
## Length:3000      Negative:1500
## Class :character      Positive:1500
## Mode  :character

str(SentimentData)

## 'data.frame':  3000 obs. of  2 variables:
## $ words      : chr  "So there is no way for me to plug it in here in the US
##                  unless I go by a converter." "Good case, Excellent value." "Great for the jaw
##                  bone." "Tied to charger for conversations lasting more than 45 minutes.MAJOR
##                  PROBLEMS!!" ...
## $ sentiment: Factor w/ 2 levels "Negative","Positive": 1 2 2 1 2 1 1 2 1
## 1 ...

#-----#

# Create a corpus (collection) of documents
corpus <- Corpus(VectorSource(SentimentData$words))

# Prepare the data for analysis (clean)
# (1) all lowercase
# (2) remove all numbers
# (3) remove all punctuation
# (4) remove all whitespaces
# (5) remove stopwords

clean <- tm_map(corpus, tolower)
clean <- tm_map(clean, removeNumbers)
clean <- tm_map(clean, removePunctuation)
clean <- tm_map(clean, removeWords, stopwords())
clean <- tm_map(clean, stripWhitespace)

#-----#
# Comparison after processing:
#-----#

```

```
inspect(corpus[1:5])

## <<SimpleCorpus>>
## Metadata: corpus specific: 1, document level (indexed): 0
## Content: documents: 5
##
## [1] So there is no way for me to plug it in here in the US unless I go by
a converter.
## [2] Good case, Excellent value.
## [3] Great for the jawbone.
## [4] Tied to charger for conversations lasting more than 45 minutes.MAJOR P
ROBLEMS!!
## [5] The mic is great.

inspect(clean[1:5])

## <<SimpleCorpus>>
## Metadata: corpus specific: 1, document level (indexed): 0
## Content: documents: 5
##
## [1] way plug us unless go converter
## [2] good case excellent value
## [3] great jawbone
## [4] tied charger conversations lasting minutesmajor problems
## [5] mic great

#-----#
# Create a matrix of document terms (words) & remove infrequent words to redu
ce number of predictors
#-----#

WordMat <- DocumentTermMatrix(clean)
freq <- findFreqTerms(WordMat, lowfreq = 5) # we only want words occuring in
at least 5 reviews
WordMat <- DocumentTermMatrix(clean, list(dictionary = freq))
inspect(WordMat[1:5,1:10])

## <<DocumentTermMatrix (documents: 5, terms: 10)>>
## Non-/sparse entries: 11/39
## Sparsity : 78%
## Maximal term length: 9
## Weighting : term frequency (tf)
## Sample :
## Terms
## Docs case charger excellent good great plug problems unless value way
## 1 0 0 0 0 0 1 0 1 0 1
## 2 1 0 1 1 0 0 0 0 1 0
## 3 0 0 0 0 0 1 0 0 0 0
## 4 0 1 0 0 0 0 0 1 0 0
## 5 0 0 0 0 0 1 0 0 0 0

#-----#
# Split the data into training and test set (70/30)
#-----#
```



```

# Raw data
n <- nrow(SentimentData)
set.seed(24)
train <- sample(1:n, n*0.7, replace=F)
raw.train <- SentimentData[train,]
raw.test  <- SentimentData[(1:n)[-train],]

# Cleaned data corpus
clean.train <- clean[train]
clean.test  <- clean[-train]

# Document term matrix
WordMat.train <- WordMat[train,]
WordMat.test  <- WordMat[-train,]

#-----#
# Word cloud to display word frequencies
#-----#

wordcloud(clean.train, random.order = F, max.words = 50, colors = matlab.like
2(n=50))

positive <- clean.train[raw.train$sentiment=="Positive"]
negative <- clean.train[raw.train$sentiment=="Negative"]

wordcloud(positive, random.order=F, max.words = 50, colors = magenta2green(n=
50))

wordcloud(negative, random.order=F, max.words = 50, colors = magenta2green(n=
50))

#-----#
# Change word counts to factors: "Yes" if word in document, "No" otherwise
#-----#

tofactors <- function(x) {
  x <- ifelse(x > 0, 1, 0)
  return(x)
}

factor.train <- apply(WordMat.train, MARGIN = 2, tofactors)
factor.test  <- apply(WordMat.test, MARGIN = 2, tofactors)

#-----#
# Fit the Naive Bayes model
#-----#

# Constructing model and making prediction
nb <- bernoulli_naive_bayes(x=factor.train, y=raw.train$sentiment)
pred <- predict(nb, factor.test)
error <- mean(pred!=raw.test$sentiment)
table(pred, raw.test$sentiment)

```

```
##
## pred      Negative Positive
## Negative   369      83
## Positive   89      359

#-----#
# Problems with zero counts in training data for a class
#-----#

pos <- which(raw.train$sentiment=="Positive")
pos.text <- factor.train[pos,]

# Determine which items in positive class have zero counts for a column
item.pos <- NULL
for(i in 1:ncol(pos.text)) {
  if(all(pos.text[,i]==0)) {
    item.pos <- c(item.pos, i)
  }
}

# For the negative class:
neg <- which(raw.train$sentiment=="Negative")
neg.text <- factor.train[neg,]

# Determine which items in negative class have zero counts for a column
item.neg <- NULL
for(i in 1:ncol(neg.text)) {
  if(all(neg.text[,i]==0)) {
    item.neg <- c(item.neg, i)
  }
}

length(item.pos)

## [1] 83

length(item.neg)

## [1] 64

# Problems - they will never classify to class if 'Yes' for that attribute
# We can use a Bayesian approach to overcome this - Laplace Smoothing

# Positive items
row.names(nb$prob1)[item.pos[1]]

## [1] "unless"

nb$prob1[item.pos[1],]

##      Negative      Positive
## 0.004798464 0.000000000

# Negative items
row.names(nb$prob1)[item.neg[1]]
```

```
## [1] "blue"

nb$prob1[item.neg[1],]

##      Negative      Positive
## 0.000000000 0.002835539

#-----#
# Laplace Smoothing - slight improvement
#-----#

nb.Laplace <- bernoulli_naive_bayes(x=factor.train, y=raw.train$sentiment, la
place = 1)
pred.Laplace <- predict(nb.Laplace, factor.test)
laplace.error <- mean(pred.Laplace!=raw.test$sentiment)
table(pred.Laplace, raw.test$sentiment)

##
## pred.Laplace Negative Positive
##      Negative      371      78
##      Positive      87     364

# Positive items
row.names(nb.Laplace$prob1)[item.pos[1]]

## [1] "unless"

nb.Laplace$prob1[item.pos[1],]

##      Negative      Positive
## 0.0057471264 0.0009433962

# Negative items
row.names(nb.Laplace$prob1)[item.neg[1]]

## [1] "blue"

nb.Laplace$prob1[item.neg[1],]

##      Negative      Positive
## 0.0009578544 0.0037735849
```

## APPENDIX C

### CHAPTER 4 CODE

```

suppressMessages(library(MASS))
suppressMessages(library(grDevices))
suppressMessages(library(mvtnorm))

#-----#
# TRUE FUNCTION AND DATA GENERATION
#-----#

# FUNCTION:  $y = -0.8x + 0.23$ 

x <- seq(from=-2, to=2, length=100)
y <- function(x) {-0.8*x+0.23}
plot(x,y(x), type='l', col="maroon", lwd=2, xlim=c(-1,1), ylim=c(-1,1), asp=1
xlab="x", ylab="y")

# Function generating the data
get.data <- function(n, tau) {
  xpt <- runif(n, min=-1, max=1)
  e <- rnorm(n, mean=0, sd=1/sqrt(tau))
  ypt <- y(xpt)+e
  out <- data.frame(x=xpt, y=ypt)
  return(out)
}

#-----#
# Plot of target function and data
#-----#

set.seed(823)
n <- 20
tau <- 16
all.data <- get.data(n, tau)
points(all.data, pch=16, xlim=c(-1,1), ylim=c(-1,1))

#-----#
# BAYESIAN APPROACH
#-----#

bayesian.regression <- function(data,tau, mu0, Sigma0, plots=F) {

  y <- data$y
  X <- cbind(1, data$x)
  seq <- seq(from=-2, to=2, length=100)
  grid <- expand.grid(seq,seq) # these points will represent all possible the
ta values

  # Plot the prior distribution
  # prior ~  $N(0, 1/4I)$  # alpha = 4
  plot.prior <- function(mean, covariance) {
    # generate sample from distribution

```

```

prior.z <- matrix(dmvnorm(x=grid, mean, covariance),100,100)

# PLOTTING
levels <- pretty(range(prior.z), 20)
filled.contour(x=x, y=x,z=prior.z, xlim=c(-1,1), ylim=c(-1,1), asp=1, add
=T,
               col = cm.colors(length(levels)-1), plot.title = title(main
="Prior"), lty=1)
}

#-----#

# Plot the likelihood function
plot.likelihood <- function(data) {

  # likelihood function
  likelihood <- function(theta) {
    tau^n*(2*pi)^(-n/2)*exp(-tau/2*t(y-X%%t(theta))%%(y-X%%t(theta)))
  }

  znum <- NULL
  for(i in 1:nrow(grid)) {
    znum[i] <- likelihood(grid[i,])
  }
  z <- matrix(data=znum, 100,100)
  levels <- pretty(range(z), 20)
  filled.contour(x=x, y=x,z=z, xlim=c(-1,1), ylim=c(-1,1), asp=1,
                col = cm.colors(length(levels)-1), plot.title = title(main
="Likelihood"))
}

#-----#

# Plot the posterior
plot.posterior <- function() {
  SigmaN <- solve(tau*t(X)%*%X+solve(Sigma0))
  muN <- SigmaN%%(tau*t(X)%*%y+solve(Sigma0)%*%mu0)
  posterior.z <- matrix(dmvnorm(x=grid, muN, SigmaN),100,100)
  levels <- pretty(range(posterior.z), 20)
  filled.contour(x=x, y=x, z=posterior.z, xlim=c(-1,1), ylim=c(-1,1), asp=1
,
                plot.title = title(main="Posterior"),
                col=cm.colors(length(levels)-1))
  return(list(muN=muN, SigmaN=SigmaN))
}

#-----#

# Predictive distribution
predictive.distribution <- function() {
  newx <- cbind(1,seq(from=-2, to=2, length=1000))

  # posterior mean and covariance
  muN <- posterior$muN
  SigmaN <- posterior$SigmaN

```

```

# predictive distribution mean and variance
pred.mean <- as.numeric(newx%*%muN)
pred.var <- 1/tau + diag(newx%*%SigmaN%*%t(newx))

# to plot the predictive distribution
plot(0, type="n", xlim=c(-1,1), ylim=c(-1,1))
polygon(x=c(newx[,2], rev(newx[,2])), y=c(pred.mean-pred.var, rev(pred.me
an+pred.var)),
        col="darkseagreen1", border=NA)
points(data, pch=20, xlim=c(-1,1), ylim=c(-1,1), asp=1)
abline(a=0.23, b=-0.8, lwd=2, col="maroon") # true regression line
points(x=newx[,2], y=pred.mean, type='l', lwd=2, col="darkgreen") # predi
ctive mean

# Include the frequentist line
#lm.fit <- lm(y~x, data=all.data)
#lm.coefs <- lm.fit$coefficients
#abline(a=lm.coefs[1], b=lm.coefs[2], col="purple", lwd=2, lty=2)
}

#-----#

# PLOTTING
if(plots==T) {
  dev.new()
  plot.prior(mu0, Sigma0) # prior
  dev.new()
  plot.likelihood(data) # Likelihood
  dev.new()
  posterior <- plot.posterior() # posterior

  # Draw samples from posterior and plot with original data
  samples <- mvrnorm(n=5, mu=posterior$muN, Sigma=posterior$SigmaN)
  dev.new()
  plot(0, type="n", xlim=c(-1,1), ylim=c(-1,1), main="Generated Samples", a
sp=1)
  for(i in 1:nrow(samples)) {
    abline(a=samples[i,1], b=samples[i,2], lwd=2, col="darkgreen")
  }
  points(data, pch=20)

  dev.new()
  predictive.distribution()
}

# Return the parameters of the posterior distribution
SigmaN <- solve(tau*t(X)%*%X+solve(Sigma0))
muN <- SigmaN%*%(tau*t(X)%*%y+solve(Sigma0)%*%mu0)
posterior <- list(Mean=muN, Covariance=SigmaN)
return(posterior)
}

```

```

#-----#
# Generate Data
#-----#

set.seed(823)
n <- 20
tau <- 16 # variance of data = 1/16
alpha <- 4 # variance of prior = 1/4
mu0 <- c(0,0)
Sigma0 <- matrix(c(1/alpha, 0, 0, 1/alpha), 2)
#Sigma0 <- matrix(c(1/30, 0, 0, 1/30), 2) # informative prior
#Sigma0 <- matrix(c(10, 0, 0, 10), 2) # non-informative prior

all.data <- get.data(n, tau)
data.1 <- all.data[1,] # only 1 observation
data.2 <- all.data[1:2,] # two observations
data.5 <- all.data[1:5,] # 5 observations
data.10 <- all.data[1:10,] # 10 (half) observations

#-----#
# Run the Bayesian analysis on the dataset containing: 1,2,5,10 and all obser
# vations
#-----#

one.out <- bayesian.regression(data=data.1, tau, mu0=mu0, Sigma0=Sigma0, plot
s=T)
two.out <- bayesian.regression(data=data.2, tau, mu0=mu0, Sigma0=Sigma0, plot
s=T)
five.out <- bayesian.regression(data=data.5, tau, mu0=mu0, Sigma0=Sigma0, plo
ts=T)
ten.out <- bayesian.regression(data=data.10, tau, mu0=mu0, Sigma0=Sigma0, plo
ts=T)
(all.out <- bayesian.regression(data=all.data, tau, mu0=mu0, Sigma0=Sigma0, p
lots=T))

## $Mean
##           [,1]
## [1,]  0.1390796
## [2,] -0.7630312
##
## $Covariance
##           [,1]      [,2]
## [1,]  0.0031494348 -0.0007329221
## [2,] -0.0007329221  0.0085245496

#-----#
# Demonstrate how the predictive variance tends to the noise
#-----#

training.sizes <- c(1,2,5,10,20,50,100,1000)
pred.var <- pred.mean <- NULL
for(i in training.sizes) {
  set.seed(823)

```

```

data.i <- get.data(n=i, tau)
out <- bayesian.regression(data=data.i,tau, mu0, Sigma0)

# posterior mean and covariance
muN <- out$Mean
SigmaN <- out$Covariance

# new sample point
set.seed(1996)
newx <- as.numeric(get.data(n=1, tau))

# mean and variance
pred.mean <- c(pred.mean, t(muN)%*%newx)
pred.var <- c(pred.var, 1/tau + diag(t(newx)%*%SigmaN%*%newx))
}

pred.var

## [1] 0.07189238 0.06767231 0.06461004 0.06392533 0.06342281 0.06281681 0.06
266156
## [8] 0.06251762

diff <- abs(pred.var-1/tau)
format(cbind(pred.var, diff), digits = 6, scientific = F)

##      pred.var      diff
## [1,] "0.0718923839" "0.0093923839"
## [2,] "0.0676723103" "0.0051723103"
## [3,] "0.0646100390" "0.0021100390"
## [4,] "0.0639253291" "0.0014253291"
## [5,] "0.0634228101" "0.0009228101"
## [6,] "0.0628168110" "0.0003168110"
## [7,] "0.0626615565" "0.0001615565"
## [8,] "0.0625176158" "0.0000176158"

pred.mean # posterior predictive mean

## [1] 0.1960867 0.1752661 0.2793209 0.2365919 0.1912898 0.2022474 0.2330972
## [8] 0.2389108

#-----#
# Comparing different model complexities
# Compare on polynomials of degree 0-10
#-----#

log.marginal.likelihood <- function(M, data, alpha, tau) {
  N <- nrow(data)
  y <- data$y

  Phi <- matrix(rep(1, N))
  # Create design matrix for polynomial function
  if(M>0) {
    for(i in 1:M) {Phi <- cbind(Phi, data$x^i)}
  }
}

```



```

# Posterior mean and covariance matrices for this design matrix
SigmaN <- solve(alpha*diag(M+1)+tau*t(Phi)%*%Phi)
muN <- tau*SigmaN%*%t(Phi)%*%y

# Compute the Log marginal Likelihood
lml <- M/2*log(alpha)+N/2*log(tau)-tau/2*t(y-Phi%*%muN)%*%(y-Phi%*%muN)-alp
ha/2*t(muN)%*%muN-
  1/2*log(det(solve(SigmaN)))-N/2*log(2*pi)
  return(lml)
}

lml <- numeric(11)
for(i in 1:11) {
  lml[i] <- log.marginal.likelihood(M=i-1, data=all.data, alpha=4, tau=16)
}

exp(lml)

## [1] 4.435767e-17 5.549252e-03 2.043188e-03 4.095125e-03 3.017783e-03
## [6] 2.583817e-03 2.171564e-03 1.875421e-03 1.693854e-03 1.528622e-03
## [11] 1.438033e-03

#-----#
plot(x=0:10, y=exp(lml), type='o', pch=16, ylab="Model Evidence", xlab="Degree Polynomial",
     main="Model Evidence", lwd=2, col="purple", xaxt="n", ylim=c(0,0.006))
axis(side = 1, at = c(0:10))
abline(v=which.max(lml)-1, lty=2, lwd=2, col="blue")
(BayesFactor12 <- exp(lml[2])/exp(lml[4]))

## [1] 1.355087

#-----#
# Increasing sample size to 50
#-----#

data.50 <- get.data(n=50, tau)

lml.50 <- numeric(11)
for(i in 1:11) {
  lml.50[i] <- log.marginal.likelihood(M=i-1, data=data.50, alpha=4, tau=16)
}
(BayesFactor12 <- exp(lml.50[2])/exp(lml.50[4]))

## [1] 7.759866

#-----#
# The EM algorithm
#-----#

alpha.em <- tau.em <- 1
X <- cbind(1,all.data$x)
y <- all.data$y
count <- 1
max.iter <- 50

```

```

eps <- 0.0001

while(TRUE) {
  Sigma <- solve(alpha.em[count]*diag(2)+tau.em[count]*t(X)%*%X)
  mu <- tau.em[count]*Sigma%*%t(X)%*%y

  alpha.em <- c(alpha.em, as.numeric(2/(sum(diag(Sigma))+t(mu)%*%mu)))
  tau.em <- c(tau.em, as.numeric(nrow(X)/(t(y-X%*%mu)%*%(y-X%*%mu)+sum(diag(X)
%*%Sigma%*%t(X)))))
  count <- count + 1
  diff <- abs(tau.em[count]-tau.em[count-1])+abs(alpha.em[count]-alpha.em[count-1])
  if(count==max.iter || diff < eps) break
}

#-----#
# Plots
#-----#

par(mfrow=c(1,2), pty="s")
plot(tau.em, type='o', pch=16, lwd=2, ylim=c(1,17), ylab=expression(tau), xlab="Iterations",
      col="darkslateblue")
abline(h=tau, lty=2, lwd=2, col="slateblue1")
plot(alpha.em, type='o', pch=16, lwd=2, ylim=c(1,4), ylab = expression(alpha),
      xlab="Iterations",
      col="darkslateblue")
abline(h=alpha, lty=2, lwd=2, col="slateblue1")

#-----#
# Using values from EM algorithm
#-----#

(alpha.tuned <- alpha.em[count])
## [1] 3.214333

(tau.tuned <- tau.em[count])
## [1] 16.50572

mu0 <- c(0,0)
Sigma0.tuned <- matrix(c(1/alpha.tuned, 0, 0, 1/alpha.tuned), 2)

full <- bayesian.regression(data=all.data, tau = tau.tuned, mu0, Sigma0.tuned,
plots=T)

#-----#
# COMPARE LARGE SAMPLE SIZE TO FREQUENTIST APPROACH
#-----#

lmfit <- lm(y~x, data = data.i)
summary(lmfit)

```

```
##
## Call:
## lm(formula = y ~ x, data = data.i)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.8393 -0.1495 -0.0084  0.1621  0.8142
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.236441   0.007568   31.24  <2e-16 ***
## x           -0.802131   0.013423  -59.76  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2392 on 998 degrees of freedom
## Multiple R-squared:  0.7816, Adjusted R-squared:  0.7814
## F-statistic: 3571 on 1 and 998 DF, p-value: < 2.2e-16

# print out values for posterior predictive distribution
muN

##              [,1]
## [1,]  0.2363687
## [2,] -0.8014962

SigmaN

##              [,1]      [,2]
## [1,] 6.256603e-05 -4.007642e-06
## [2,] -4.007642e-06  1.967040e-04
```

## APPENDIX D

### CHAPTER 5 CODE

```

#-----#
# Spike and slab prior plots
#-----#

x <- seq(from=-10, to=10, length=10000)

par(mfrow=c(1,2),pty='s', mar=c(4,3,3,1))

#-----#
# Plot the Dirac Spike
#-----#

plot(x, y=rep(0.1,10000), type='l', ylim=c(0,5), yaxt='n',ylab = "", xlab="",
col="blue", lwd=2,
      main="Dirac Spike")
abline(v=0, lty=2, col="red", lwd=2)

#-----#
# Plot the continuous spike
#-----#

spike <- dnorm(x, mean=0, sd=0.3)
slab <- dnorm(x, mean=0, sd=4)
plot(x, spike, type='l',col="red", ylim=c(0,1.5), yaxt='n', ylab="", lwd=2,
      main="Absolutely Continuous Spike")
points(x, slab, type = 'l', col="blue", ylim=c(0,1.5), lwd=2)

#-----#
# PARADOXES
#-----#

#-----#
# Function to determine the maringal Likelihood
# Requires specific values of g, p, R2 and N
#-----#

ml <- function(g, p, R2, N) {
  ((1+g)^((N-p-1)/2))*((1+g*(1-R2))^(-(N-1)/2))
}

par(mfrow=c(2,1), mar=c(4,4,2,4))

#-----#
# Lindleys paradox # g to infinity
#-----#

N <- 30
p <- 3 # 3 predictors
R2 <- 0.8

```

```

# sequence of random g values from 0 to 1000
gseq <- seq(from=0, to=1000)
Lindley <- NULL
for(i in 1:length(gseq)) {
  Lindley <- c(Lindley, ml(gseq[i],p,R2,N))
}

plot(Lindley, type='l', lwd=2, xlab = "g values", ylab="Bayes Factor")
abline(h=0, lty=2, col="gray", lwd=2)

#-----#
# Using the EB approach:
#-----#
Fk <- (R2/p)/((1-R2)/(N-p-1))

#-----#
# Compare to maximum found
#-----#
(Fk-1)

## [1] 33.66667
which.max(Lindley)

## [1] 35

# Add to plot
abline(v=Fk-1, lwd=2, col="purple")

#-----#
# Information paradox # g to infinity
#-----#

g <- 2
# constant to converge to
cc <- (1+g)^(0.5*(N-p-1))

# sequence of random g values from 0 to 1000
R2seq <- seq(from=0, to=1, length=1000)
Info <- NULL
for(i in 1:length(R2seq)) {
  Info <- c(Info, ml(g,p,R2seq[i],N))
}

plot(R2seq, Info, type='l', lwd=2, xlab = "R-squared", ylab="Bayes Factor")
abline(h=cc, lty=2, col="gray", lwd=2)

#-----#
# VARIABLE SELECTION COMPARISON
#-----#

suppressMessages(library(ElemStatLearn))
data("prostate")
head(prostate) # Response is Lpsa

```

```
##      lcavol  lweight age      lbph svi      lcp gleason pgg45      lpsa
## 1 -0.5798185 2.769459 50 -1.386294 0 -1.386294      6      0 -0.4307829
## 2 -0.9942523 3.319626 58 -1.386294 0 -1.386294      6      0 -0.1625189
## 3 -0.5108256 2.691243 74 -1.386294 0 -1.386294      7     20 -0.1625189
## 4 -1.2039728 3.282789 58 -1.386294 0 -1.386294      6      0 -0.1625189
## 5  0.7514161 3.432373 62 -1.386294 0 -1.386294      6      0  0.3715636
## 6 -1.0498221 3.228826 50 -1.386294 0 -1.386294      6      0  0.7654678
##   train
## 1  TRUE
## 2  TRUE
## 3  TRUE
## 4  TRUE
## 5  TRUE
## 6  TRUE

prostate.scaled <- cbind(scale(prostate[,1:8]), 'lpsa'=prostate[,9])

#-----#
# Training and Test Sets
#-----#

train <- as.data.frame(prostate.scaled[prostate$train=="TRUE",])
test  <- as.data.frame(prostate.scaled[prostate$train=="FALSE",])

#-----#
# Fit a linear model (frequentist approach)
#-----#

lm.fit <- lm(lpsa~., data=train)
summary(lm.fit)

##
## Call:
## lm(formula = lpsa ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64870 -0.34147 -0.05424  0.44941  1.48675
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.46493    0.08931  27.598 < 2e-16 ***
## lcavol         0.67953    0.12663   5.366 1.47e-06 ***
## lweight        0.26305    0.09563   2.751  0.00792 **
## age           -0.14146    0.10134  -1.396  0.16806
## lbph           0.21015    0.10222   2.056  0.04431 *
## svi            0.30520    0.12360   2.469  0.01651 *
## lcp           -0.28849    0.15453  -1.867  0.06697 .
## gleason       -0.02131    0.14525  -0.147  0.88389
## pgg45          0.26696    0.15361   1.738  0.08755 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7123 on 58 degrees of freedom
```

```
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6522
## F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12

lm.coefs <- lm.fit$coefficients
lm.pred <- predict(lm.fit, newdata=test[, -9])
(lm.error <- mean((lm.pred-test$lpsa)^2))

## [1] 0.521274

(lm.stderr <- sqrt(var((lm.pred-test$lpsa)^2)/nrow(test)))

## [1] 0.178724

#-----#
# g-Prior
#-----#

suppressMessages(library(BayesVarSel))

# Compare models using (1) g=N and (2) Zellner-slow inverse gamma
gPriorSummary <- function(model.out) {
  gPrior.out <- summary(model.out)
  gPrior.coefsKEEP <- which(model.out$HPMbin==1)

  coefs <- BMACoeff(model.out)
  coefs.out <- apply(coefs, 2, median)

  par(mfrow=c(3,3), mar=c(4,3,3,1), pty='s')
  cred.out <- NULL
  for(i in 1:ncol(train)) {
    histBMA(BMACoeff(model.out), dimnames(coefs)[[2]][i], text=T)
    cred.out <- rbind(cred.out, quantile(coefs[,i], probs=c(0.025,0.975)))
  }

  # Fit the new model using these coefficients
  pred.gPrior <- predict(model.out, newdata=test[, -9])
  gPrior.medians <- apply(pred.gPrior, 2, median)
  gPrior.err <- mean((gPrior.medians-test[,9])^2)
  gPrior.stderr <- sqrt(var((gPrior.medians-test[,9])^2)/nrow(test))
  return(list(coefs=coefs.out, err=gPrior.err, stderr=gPrior.stderr, cred=round(cred.out,4)))
}

#-----#
# (1) g=N
#-----#

set.seed(24)
gN <- Bvs(lpsa~., data=train, prior.betas = "gZellner", prior.models = "Constant", n.keep = 1000)

## Info. . . .
## Most complex model has 9 covariates
## From those 1 is fixed and we should select from the remaining 8
## lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45
```

```

## The problem has a total of 256 competing models
## Of these, the 256 most probable (a posteriori) are kept
## Working on the problem...please wait.

(gN.out <- gPriorSummary(gN))

##
## Inclusion Probabilities:
##      Incl.prob.  HPM  MPM
## lcavol          1    *   *
## lweight        0.883  *   *
## age            0.1617
## lbph           0.4271
## svi            0.5435      *
## lcp            0.1889
## gleason        0.1366
## pgg45          0.3062
## ---
## Code: HPM stands for Highest posterior Probability Model and
##      MPM for Median Probability Model.
##

## $coefs
##      Intercept      lcavol      lweight      age      lbph      svi      l
cp
## 2.47152722 0.66565517 0.28200167 0.00000000 0.00000000 0.08455324 0.000000
00
##      gleason      pgg45
## 0.00000000 0.00000000
##
## $err
## [1] 0.4449182
##
## $stderr
## [1] 0.1386542
##
## $cred
##      2.5%  97.5%
## [1,] 2.2873 2.6573
## [2,] 0.4083 0.9134
## [3,] 0.0000 0.4843
## [4,] -0.2063 0.0087
## [5,] 0.0000 0.3978
## [6,] 0.0000 0.4786
## [7,] -0.3587 0.0293
## [8,] -0.0690 0.1699
## [9,] 0.0000 0.3499

#-----#
# (2) ZS prior
#-----#

set.seed(24)

```



```

gZS <- Bvs(lpsa~., data=train, prior.betas = "ZellnerSiow", prior.models = "Constant", n.keep = 1000)

## Info. . . .
## Most complex model has 9 covariates
## From those 1 is fixed and we should select from the remaining 8
## lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45
## The problem has a total of 256 competing models
## Of these, the 256 most probable (a posteriori) are kept
## Working on the problem...please wait.

(gZS.out <- gPriorSummary(gZS))

##
## Inclusion Probabilities:
##      Incl.prob.  HPM  MPM
## lcavol          1    *   *
## lweight      0.8852  *   *
## age           0.1961
## lbph          0.4608
## svi           0.5726      *
## lcp           0.2303
## gleason       0.1623
## pgg45         0.3422
## ---
## Code: HPM stands for Highest posterior Probability Model and
## MPM for Median Probability Model.
##
## Simulations obtained using the best 256 models
## that accumulate 1 of the total posterior probability
##
## $coefs
## Intercept    lcavol    lweight      age      lbph      svi      lcp    gl
## eason
## 2.4712557 0.6652504 0.2777942 0.0000000 0.0000000 0.1185711 0.0000000 0.00
## 00000
##      pgg45
## 0.0000000
##
## $err
## [1] 0.4479767
##
## $stderr
## [1] 0.1407222
##
## $cred
##      2.5%  97.5%
## [1,] 2.2870 2.6513
## [2,] 0.4061 0.9152
## [3,] 0.0000 0.4812
## [4,] -0.2262 0.0235
## [5,] 0.0000 0.3966
## [6,] 0.0000 0.4879
## [7,] -0.3948 0.0275

```

```
## [8,] -0.1056 0.1862
## [9,] 0.0000 0.3681

#-----#
# Spike and Slab Prior
#-----#

suppressMessages(library(BoomSpikeSlab))
set.seed(24)
prior <- SpikeSlabPrior(as.matrix(cbind(1,train[,1:8])), as.matrix(train[,9])
,
                        expected.model.size = 2, diagonal.shrinkage = 0,
                        optional.coefficient.estimate = rep(0, 9),)
SSPrior <- lm.spike(lpsa~., data=train, niter=10000, prior=prior)

#-----#
summary(SSPrior, burn = 1000)

## coefficients:
##              mean      sd mean.inc sd.inc inc.prob
## lcavol         0.778000 0.10800   0.7780 0.1080 1.00000
## (Intercept)    2.480000 0.09640   2.4800 0.0964 1.00000
## lweight        0.171000 0.17000   0.3150 0.0889 0.54500
## lbph           0.014900 0.06480   0.2570 0.1020 0.05780
## svi            0.006060 0.04210   0.2350 0.1220 0.02580
## pgg45          0.001970 0.02140   0.1660 0.1070 0.01190
## gleason        0.000454 0.01030   0.0716 0.1090 0.00633
## age            -0.000224 0.00936  -0.0429 0.1240 0.00522
## lcp            -0.000225 0.00834  -0.0722 0.1330 0.00311
##
## residual.sd =
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.5729 0.7276 0.7778 0.7828 0.8311 1.1592
##
## r-square      =
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07888 0.52649 0.58526 0.57586 0.63714 0.77501

par(mfrow=c(1,1), pty='m')
plot(SSPrior, burn = 1000)

plot.ts(SSPrior$beta)

cred.SS <- NULL
par(mfrow=c(2,2))
for(i in 1:ncol(SSPrior$beta)) {
  hist(SSPrior$beta[1001:10000,i], main=colnames(train)[i], xlab="Coefficient
")
  cred.SS <- rbind(cred.SS, quantile(SSPrior$beta[1001:10000,i], probs=c(0.02
5,0.975)))
}

#-----#
```

```

SS.coefs <- apply(SSPrior$beta, 2, median)
round(SS.coefs,4)

## (Intercept)      lcavol      lweight      age      lbph      svi
##      2.4777      0.7788      0.1958      0.0000      0.0000      0.0000
##      lcp      gleason      pgg45
##      0.0000      0.0000      0.0000

SSpred <- predict(SSPrior, newdata=test[, -9])
(SS.err <- mean((test[,9]-apply(SSpred,1,median))^2))

## [1] 0.4732994

(SS.stderr <- sqrt(var((apply(SSpred,1,median)-test$lpsa)^2)/nrow(test)))

## [1] 0.1363112

round(cred.SS,4)

##      2.5%  97.5%
## [1,] 2.2864 2.6641
## [2,] 0.5639 0.9892
## [3,] 0.0000 0.4632
## [4,] 0.0000 0.0000
## [5,] 0.0000 0.2780
## [6,] 0.0000 0.0076
## [7,] 0.0000 0.0000
## [8,] 0.0000 0.0000
## [9,] 0.0000 0.0000

#-----#
# Bayesian Ridge and LASSO
#-----#

suppressMessages(library(monomvn))

#-----#
# Plot the contours for Gaussian vs Laplace
#-----#

# Gaussian
x <- seq(from=-2,2,length=100)
grid <- expand.grid(x,x)
zGauss <- matrix(data=c(dmvnorm(grid, mean=c(0,0), sigma=matrix(c(1/4,0,0,1/4),2,2))),100,100)
filled.contour(x=x,y=x, z=zGauss,asp=1)

# Laplace
suppressMessages(library(nimble))

ZLap <- matrix(data=c(ddexp(grid[,1])*ddexp(rev(grid[,2]))),100,100)
filled.contour(x=x,y=x,z=ZLap, asp=1)

#-----#
# RUN TWICE (1) With RJ MCMC (2) No RJ MCMC
#-----#

```

```

suppressMessages(library(monomvn))
bridge.err <- bridge.stderr <- NULL
blasso.err <- blasso.stderr <- NULL
ridge.coefs.median <- lasso.coefs.median <- NULL
cred.ridge <- cred.lasso <- NULL
ridge.col <- c("red", "pink")
lasso.col <- c("blue", "lightblue")
j <- 1
for(RJ in c(T,F)) {

  set.seed(24)
  ridge <- bridge(X=train[,1:8], y=train[,9], T = 10000, RJ=RJ)
  ridge.out <- summary(ridge, burnin = 1000)

  plot(ridge, burnin = 1000)

  ridge.coefs <- ridge.out$coef
  ridge.coefs.median <- cbind(ridge.coefs.median, as.numeric(gsub("Median :",
"", ridge.coefs[3,])))
  cred.ridge <- rbind(cred.ridge, quantile(ridge$mu[1001:10000], probs=c(0.02
5, 0.975)))

  par(mfrow=c(4,2))
  for(i in 1:8) {
    hist(ridge$beta[1001:10000,i], main=colnames(train)[i],
        xlim=c(-0.4,0.8), xlab="", breaks=20, col=ridge.col[j])
    cred.ridge <- rbind(cred.ridge, quantile(ridge$beta[1001:1000,i],probs=c(
0.025,0.975)))
  }

#-----#
# LASSO
#-----#

  set.seed(24)
  lasso <- blasso(X=train[,1:8], y=train[,9], T=10000, RJ=RJ)
  par(mfrow=c(1,1))
  plot(lasso, burnin = 1000)
  lasso.out <- summary(lasso, burnin = 1000)
  lasso.coefs <- lasso.out$coef
  lasso.coefs.median <- cbind(lasso.coefs.median, as.numeric(gsub("Median :",
"", lasso.coefs[3,])))
  cred.lasso <- rbind(cred.lasso, quantile(lasso$mu[1001:10000], probs=c(0.02
5, 0.975)))

  par(mfrow=c(4,2))
  for(i in 1:8) {
    hist(lasso$beta[1001:10000,i], main=colnames(train)[i], xlab="", breaks =
20,
        xlim=c(-0.4,0.8), col=lasso.col[j])
    cred.lasso <- rbind(cred.lasso, quantile(lasso$beta[1001:1000,i],probs=c(
0.025,0.975)))
  }
}

```

```

# TO OBTAIN TEST ERRORS and STDERR
test.err <- function(coefs) {
  yhat <- cbind(1,as.matrix(test[, -9]))%*%coefs
  testerr <- mean((yhat-test[,9])^2)
  stderr <- sqrt(var((yhat-test[,9])^2)/nrow(test))
  return(list(testerr, stderr))
}

#-----#
# Compare in a single plot
#-----#

par(mfrow=c(1,1))
boxplot(ridge$beta[1001:10000, 1],
        lasso$beta[1001:10000, 1],
        ridge$beta[1001:10000, 2],
        lasso$beta[1001:10000, 2],
        ridge$beta[1001:10000, 3],
        lasso$beta[1001:10000, 3],
        ridge$beta[1001:10000, 4],
        lasso$beta[1001:10000, 4],
        ridge$beta[1001:10000, 5],
        lasso$beta[1001:10000, 5],
        ridge$beta[1001:10000, 6],
        lasso$beta[1001:10000, 6],
        ridge$beta[1001:10000, 7],
        lasso$beta[1001:10000, 7],
        ridge$beta[1001:10000, 8],
        lasso$beta[1001:10000, 8],
        col=rep(c(ridge.col[j],lasso.col[j]),times=8),xaxt='n')
abline(h=0, lty=2, col="gray")
for(i in seq(from=2.5, to=14.5, by=2)) {
  abline(v=i)
}
axis(1, at=seq(from=1.5, to=15.5, by=2), labels = names(lm.coefs)[-1])
legend('topright', pch=15, col=c(ridge.col[j],lasso.col[j]), legend=c("B-Ridge", "B-LASSO"))

#-----#

bridge.err <- c(bridge.err, test.err(ridge.coefs.median[,j]))[[1]]
blasso.err <- c(blasso.err, test.err(lasso.coefs.median[,j]))[[1]]
bridge.stderr <- c(bridge.stderr, test.err(ridge.coefs.median[,j]))[[2]]
blasso.stderr <- c(blasso.stderr, test.err(lasso.coefs.median[,j]))[[2]]

j <- 2
}

round(cred.ridge,4)

##          2.5%   97.5%
## [1,]  2.2831  2.6469
## [2,]  0.4767  0.7407

```

```
## [3,] 0.1160 0.3505
## [4,] 0.0000 0.0000
## [5,] 0.1487 0.1661
## [6,] 0.0227 0.3160
## [7,] 0.0000 0.0000
## [8,] -0.1097 -0.0028
## [9,] 0.0000 0.0000
## [10,] 2.2931 2.6385
## [11,] 0.4505 0.7362
## [12,] 0.1196 0.2442
## [13,] -0.1779 0.0918
## [14,] 0.0497 0.0539
## [15,] 0.2143 0.2561
## [16,] -0.2799 -0.2219
## [17,] -0.1232 -0.0917
## [18,] 0.4157 0.4671
```

```
round(cred.lasso,4)
```

```
##          2.5%  97.5%
## [1,] 2.2781 2.6568
## [2,] 0.4195 0.6920
## [3,] 0.3295 0.3438
## [4,] 0.0000 0.0000
## [5,] 0.0000 0.0000
## [6,] 0.0033 0.1298
## [7,] -0.0227 0.0677
## [8,] -0.0640 -0.0016
## [9,] 0.0525 0.1757
## [10,] 2.2789 2.6571
## [11,] 0.4839 0.5873
## [12,] 0.2432 0.2661
## [13,] -0.1446 0.0131
## [14,] 0.1192 0.2503
## [15,] 0.2871 0.3598
## [16,] -0.1798 -0.1567
## [17,] -0.0743 0.0881
## [18,] 0.1133 0.2332
```

```
#-----#
# Compare all coefficients
#-----#
```

```
AllCoefs <- cbind(lm.coefs, gN.out$coefs, gZS.out$coefs, SS.coefs, ridge.coefs
  .median[,1],
  ridge.coefs.median[,2], lasso.coefs.median[,1], lasso.coefs
  .median[,2])
```

```
#-----#
# Compare all Errors
#-----#
```

```
AllErrors <- rbind(lm.error, gN.out$err, gZS.out$err, SS.err,
  bridge.err[1], bridge.err[2],
  blasso.err[1], blasso.err[2])
```

```

AllStdErrs <- rbind(lm.stderr, gN.out$stderr, gZS.out$stderr, SS.stderr,
                   bridge.stderr[1], bridge.stderr[2],
                   blasso.stderr[1], blasso.stderr[2])
Compare <- data.frame(AllErrors, AllStdErrs)
rownames(Compare) <- colnames(AllCoefs) <- c("Full Model", "gPrior-Zellner", "g
Prior-ZellnerSioW",
                                             "Spike-and-Slab", "B-Ridge: RJ",
                                             "B-Ridge", "B-LASSO: RJ",
                                             "B-LASSO")
colnames(Compare) <- c("Test Error", "Std Error")
round(Compare, 4)

##                Test Error Std Error
## Full Model      0.5213    0.1787
## gPrior-Zellner   0.4449    0.1387
## gPrior-ZellnerSioW 0.4480    0.1407
## Spike-and-Slab   0.4733    0.1363
## B-Ridge: RJ      0.4323    0.1278
## B-Ridge          0.4920    0.1626
## B-LASSO: RJ      0.4465    0.1407
## B-LASSO          0.4546    0.1483

round(AllCoefs, 4)

##                Full Model gPrior-Zellner gPrior-ZellnerSioW Spike-and-Slab
## (Intercept)      2.4649      2.4715      2.4713      2.4777
## lcavol            0.6795      0.6657      0.6653      0.7788
## lweight           0.2631      0.2820      0.2778      0.1958
## age              -0.1415      0.0000      0.0000      0.0000
## lbph              0.2101      0.0000      0.0000      0.0000
## svi               0.3052      0.0846      0.1186      0.0000
## lcp               -0.2885      0.0000      0.0000      0.0000
## gleason           -0.0213      0.0000      0.0000      0.0000
## pgg45             0.2670      0.0000      0.0000      0.0000
##                B-Ridge: RJ B-Ridge B-LASSO: RJ B-LASSO
## (Intercept)      2.4680  2.4690      2.4680  2.4680
## lcavol            0.5891  0.5662      0.5809  0.5565
## lweight           0.2642  0.2572      0.2358  0.2266
## age              0.0000 -0.1082      0.0000 -0.0429
## lbph              0.1013  0.1965      0.0780  0.1457
## svi               0.2162  0.2786      0.1578  0.2052
## lcp               0.0000 -0.1512      0.0000 -0.0345
## gleason           0.0000  0.0190      0.0000  0.0175
## pgg45             0.0000  0.1921      0.0000  0.1093

#-----#
# SVMs
#-----#

suppressMessages(library(mvtnorm))
suppressMessages(library(e1071))

mu1 <- c(1, 1)
mu2 <- c(-1, -1)
Sigma1 <- matrix(c(1, -0.5, -0.5, 1), 2, 2)

```

```

Sigma2 <- matrix(c(0.5,-0.1,-0.1,0.5),2,2)

set.seed(6)
Blue <- rmvnorm(60, mu1, Sigma1)
Red <- rmvnorm(60, mu2, Sigma2)

#-----#
# SET UP DATA AND GRID
#-----#

y <- c(rep(1, 60), rep(-1,60))
X <- rbind(Blue,Red)
Data <- data.frame(X, y=as.factor(y))
x <- seq(from=-3,to=3, length=250)
grid <- expand.grid(X1=x,X2=x)

#-----#
# FIT THE OSH, SVC AND SVM
#-----#

OSH <- svm(formula = y~., data=Data, kernel = 'linear', scale = F, cost=100)
SVC <- svm(formula = y~., data=Data, kernel = 'linear', scale = F, cost=1)
SVM <- svm(formula = y~., data=Data, kernel = 'radial', scale = F, cost=1)

#-----#
# PLOT THE OSH and SVC
#-----#

svmplot <- function(svmfit) {
  par(pty='s')
  supportvecs <- svmfit$SV
  svmbeta <- t(svmfit$coefs)%*%svmfit$SV
  svmbeta0 <- svmfit$rho

  plot(0, type='n', xlim=c(-3,3), ylim=c(-3,3), axes = F)
  grid.pred <- predict(svmfit, newdata=grid)

  # Shade the entire grid
  points(grid, col=ifelse(grid.pred==1, "paleturquoise1","plum1"), pch=16, cex=2)

  # Decision boundary and margins
  abline((svmbeta0) / svmbeta[2], -svmbeta[1] / svmbeta[2], lwd=3)
  abline((svmbeta0 - 1) / svmbeta[2], -svmbeta[1] / svmbeta[2], lwd=2, col="gray")
  abline((svmbeta0 + 1) / svmbeta[2], -svmbeta[1] / svmbeta[2], lwd=2, col="gray")

  # Add the data points
  points(Blue, pch=16, col="blue", xlim=c(-3,3), ylim=c(-3,3))
  points(Red, pch=16, col="maroon", xlim=c(-3,3), ylim=c(-3,3))
  points(supportvecs, pch=0, cex=2) # support vectors
}

```



```

#-----#
# Plot the OSH and SVC
#-----#

svmplot(OSH)
svmplot(SVC)

#-----#
# PLOT THE SVM
#-----#

grid.pred <- predict(SVM, newdata=grid, decision.values = T, method="class")
z <- attributes(grid.pred)$decision
plot(0, type='n', xlim=c(-3,3), ylim=c(-3,3), axes = F)
# Shade the entire grid
points(grid, col=ifelse(grid.pred==1, "paleturquoise1", "plum1"), pch=16)

#-----#
# Decision boundary and margin
#-----#

contour(x,x,matrix(z, 250,250), levels=0, drawlabels = F, axes=FALSE, lwd=2,
col="black", add=T)
contour(x,x,matrix(z, 250,250), levels=-1, drawlabels = F, axes=FALSE, lwd=2,
col="gray", add=T)
contour(x,x,matrix(z, 250,250), levels=1, drawlabels = F, axes=FALSE, lwd=2,
col="gray", add=T)

#-----#
# Add the data points
#-----#

points(Blue, pch=16, col="blue", xlim=c(-3,3), ylim=c(-3,3))
points(Red, pch=16, col="maroon", xlim=c(-3,3), ylim=c(-3,3))
points(SVM$SV, pch=0, cex=2) # support vectors

#-----#
# RVM vs SVM
#-----#

suppressMessages(library(mvtnorm))
suppressMessages(library(e1071))
suppressMessages(library(kernlab))

#-----#
# REGRESSION
#-----#

# Generate the noisy data
x <- seq(from=-5,to=5, length=1000)
y <- sin(3*x)+sin(-x)

set.seed(1)
sample <- sample(1:1000, size=50, replace=T)
sampleX <- x[sample]

```

```

sampley <- y[sample]+rnorm(50, sd=0.5)

Data <- data.frame(X=sampleX, y=sampley)

#-----#
# SUPPORT VECTOR REGRESSION
#-----#

SVMReg <- svm(formula=y~X, data=Data, scale=F)
plot(Data, pch=16, col="maroon", xlim=c(-4.5,4.5), ylim=c(-4.5,4.5),
      xlab="", ylab="")
grid.data <- data.frame(X=x)
SVMReg.pred <- predict(SVMReg, newdata=grid.data, interval="pred")
lines(x,y, type='l', asp=1, lwd=2, xlim=c(-4.5,4.5), ylim=c(-4.5,4.5))
lines(x, SVMReg.pred, lwd=2, col="lightslateblue")

#-----#
# Obtain the support vectors
#-----#

svmSV <- SVMReg$SV
length(svmSV) # number of support vectors

## [1] 46

sv <- NULL
for(i in 1:length(svmSV)) {
  sv <- c(sv, which(svmSV[i]==Data$X))
}

# plot a box around the support vectors
points(Data[sv, ], pch=0, cex=2)

#-----#
# RELEVANCE VECTOR REGRESSION
#-----#

plot(Data, pch=16, col="maroon", xlim=c(-4.5,4.5), ylim=c(-4.5,4.5),
      xlab="", ylab="")
lines(x,y, type='l', asp=1, lwd=2, xlim=c(-4.5,4.5), ylim=c(-4.5,4.5))
set.seed(24)
RVMReg <- rvm(x=Data$X, y=Data$y)

## Using automatic sigma estimation (sigest) for RBF or laplace kernel

RVMRegPred <- predict(RVMReg, newdata=grid.data)
lines(x, RVMRegPred, lwd=2, col="darkolivegreen3")
RVs <- RVMReg@RVindex # Index of the relevance vectors
length(RVs) # number of relevance vectors

## [1] 8

# draw a box around the relevance vectors
points(Data[RVs, ], pch=0, cex=2)

```

```

#-----#
# CLASSIFICATION
#-----#

suppressMessages(library(mvtnorm))
suppressMessages(library(e1071))

#-----#
# SET UP DATA AND GRID
# SIMILAR POINTS TO BEFORE, EXCEPT NOW THE CLASSES OVERLAP
#-----#

mu1 <- c(1,1)
mu2 <- c(-1,-1)
Sigma1 <- matrix(c(2,-1,-1,2),2,2)
Sigma2 <- matrix(c(1.5,-0.8,-0.8,2),2,2)

set.seed(143)
Blue <- rmvnorm(60, mu1, Sigma1)
Red <- rmvnorm(60, mu2, Sigma2)

y <- c(rep(1, 60), rep(-1,60))
X <- rbind(Blue,Red)
Data <- data.frame(X, y=as.factor(y))
x <- seq(from=-4.5,to=4.5, length=250)
grid <- expand.grid(X1=x,X2=x)

#-----#
# FIT THE SUPPORT VECTOR MACHINE
#-----#

SVM <- svm(formula = y~., data=Data, kernel = 'radial', scale = F, cost=20)

#-----#
# PLOT THE SVM
#-----#

grid.pred <- predict(SVM, newdata=grid, decision.values = T, method="class")
z <- attributes(grid.pred)$decision
par(pty='s')
plot(0, type='n', xlim=c(-4.5,4.5), ylim=c(-4.5,4.5), axes = F)

# Shade the entire grid
points(grid, col=ifelse(grid.pred==1, "paleturquoise1","plum1"), pch=16)

# Decision boundary and margin
contour(x,x,matrix(z, 250,250), levels=0, drawlabels = F, axes=FALSE, lwd=2,
col="black", add=T)

# Add the data points
points(Blue, pch=16, col="blue", xlim=c(-4.5,4.5), ylim=c(-4.5,4.5))
points(Red, pch=16, col="maroon", xlim=c(-4.5,4.5), ylim=c(-4.5,4.5))
points(SVM$SV, pch=0, cex=2) # support vectors

```

```

length(SVM$SV)
## [1] 54

#-----#
# FIT AND PLOT AN RVM
#-----#

RVMclass <- rvm(X,y)

## Using automatic sigma estimation (sigest) for RBF or laplace kernel

# Predict the grid of points
RVMgrid.pred <- predict(RVMclass, newdata=grid)
z <- ifelse(RVMgrid.pred > 0, "paleturquoise1", "plum1")
plot(0, type='n', xlim=c(-4.5,4.5), ylim=c(-4.5,4.5), axes = F)

# Shade the grid
points(grid, col=z, pch=16)

# Plot the data
points(Blue, pch=16, col="blue", xlim=c(-4.5,4.5), ylim=c(-4.5,4.5))
points(Red, pch=16, col="maroon", xlim=c(-4.5,4.5), ylim=c(-4.5,4.5))

# Plot the decision boundary
contour(x,x,matrix(RVMgrid.pred, 250,250), levels=0, drawlabels = F, axes=FALSE, lwd=2, col="black", add=T)

# Number of support vectors
length(RVMclass@RVindex)

## [1] 12

points(Data[RVMclass@RVindex,], pch=0, cex=2)

#-----#
# GENERATE TEST DATA AND COMPARE THE PERFORMANCE
#-----#

set.seed(1212)
Blue <- cbind(rmvnorm(100, mu1, Sigma1),1)
Red <- cbind(rmvnorm(100, mu2, Sigma2),-1)

TestData <- rbind(Blue, Red)
TestData <- TestData[sample(1:200), ]

# SVM test error
SVMtest <- predict(SVM, newdata=TestData[,1:2])
mean(SVMtest!=TestData[,3])

## [1] 0.135

# RVM test error
RVMtest <- predict(RVMclass, newdata=TestData[,1:2])
RVMtest <- ifelse(RVMtest < 0, -1,1)
mean(RVMtest!=TestData[,3])

```

```
## [1] 0.09
```

## APPENDIX E

### CHAPTER 6 CODE

```

suppressMessages(library(bnlearn))
par(mfrow=c(1,2))

#-----#
# Fully connected graph
#-----#

# Create the nodes
example5 = empty.graph(nodes = c("x1", "x2", "x3", "x4", "x5"))

# Define all the edges
arc.set = matrix(c("x1", "x2",
                   "x1", "x3",
                   "x2", "x3",
                   "x1", "x4",
                   "x2", "x4",
                   "x3", "x4",
                   "x1", "x5",
                   "x2", "x5",
                   "x3", "x5",
                   "x4", "x5"),
                 byrow = TRUE, ncol = 2, dimnames = list(NULL, c("from", "to"
)))
arcs(example5) = arc.set
graphviz.plot(example5) # plot

## Loading required namespace: Rgraphviz

#-----#
# Example 5 - conditional independencies
#-----#

# Define the new edges
arc.set = matrix(c("x1", "x5",
                   "x2", "x5",
                   "x1", "x4",
                   "x1", "x3",
                   "x3", "x5"),
                 byrow = TRUE, ncol = 2, dimnames = list(NULL, c("from", "to"
)))
arcs(example5) = arc.set
graphviz.plot(example5)

#-----#
# Naive Bayes
# great, good, movie, bad, phone
#-----#

par(mfrow=c(1,1))

```

```

# Create the nodes
naive = empty.graph(nodes = c("Sentiment", "great", "good", "movie", "bad", "phone"))

# Define the edges
arc.set = matrix(c("Sentiment", "great",
                   "Sentiment", "good",
                   "Sentiment", "movie",
                   "Sentiment", "bad",
                   "Sentiment", "phone"),
                 byrow = TRUE, ncol = 2, dimnames = list(NULL, c("from", "to")))
arcs(naive) = arc.set
graphviz.plot(naive) # plot

#-----#
# Load the asia dataset from bnlearn package
#-----#

data(asia)
dat <- asia
attach(dat)

## The following object is masked from package:base:
##
##      T

#-----#
# create and plot the network structure.
#-----#

modelstring = paste0("[Asia][Smoking][TB|Asia][LungCancer|Smoking][Bronchitis|Smoking][Dyspnoea|Bronchitis:Either][Either|TB:LungCancer][Xray|Either]")
dag <- model2network(modelstring)
graphviz.plot(dag, layout = "dot")

#-----#
# Fit the parameters of the BN
#-----#

colnames(dat) <- c("Asia", "Smoking", "TB", "LungCancer", "Bronchitis", "Either", "Xray", "Dyspnoea")
bnfit <- bn.fit(x=dag, data=dat, method = "mle")
bnfit

##
##   Bayesian network parameters
##
##   Parameters of node Asia (multinomial distribution)
##
##   Conditional probability table:
##       no    yes
## 0.9916 0.0084
##
##   Parameters of node Bronchitis (multinomial distribution)

```

```

##
## Conditional probability table:
##
##           Smoking
## Bronchitis   no      yes
##           no 0.7006036 0.2823062
##           yes 0.2993964 0.7176938
##
## Parameters of node Dyspnoea (multinomial distribution)
##
## Conditional probability table:
##
## , , Either = no
##
##           Bronchitis
## Dyspnoea     no      yes
##           no 0.90017286 0.21373057
##           yes 0.09982714 0.78626943
##
## , , Either = yes
##
##           Bronchitis
## Dyspnoea     no      yes
##           no 0.27737226 0.14592275
##           yes 0.72262774 0.85407725
##
##
## Parameters of node Either (multinomial distribution)
##
## Conditional probability table:
##
## , , TB = no
##
##           LungCancer
## Either no yes
##       no  1  0
##       yes 0  1
##
## , , TB = yes
##
##           LungCancer
## Either no yes
##       no  0  0
##       yes 1  1
##
##
## Parameters of node LungCancer (multinomial distribution)
##
## Conditional probability table:
##
##           Smoking
## LungCancer   no      yes
##           no 0.98631791 0.88230616
##           yes 0.01368209 0.11769384

```



```
##
## Parameters of node Smoking (multinomial distribution)
##
## Conditional probability table:
##   no   yes
## 0.497 0.503
##
## Parameters of node TB (multinomial distribution)
##
## Conditional probability table:
##
##      Asia
## TB      no      yes
##  no  0.991528842 0.952380952
##  yes 0.008471158 0.047619048
##
## Parameters of node Xray (multinomial distribution)
##
## Conditional probability table:
##
##      Either
## Xray      no      yes
##  no  0.956587473 0.005405405
##  yes 0.043412527 0.994594595

#-----#
# Simple inference example
#-----#
two = empty.graph(nodes = c("x","y")) # two nodes

# Define the edge
arc.set = matrix(c("x", "y"),
                  byrow = TRUE, ncol = 2, dimnames = list(NULL, c("from", "to"
)))
arcs(two) = arc.set
graphviz.plot(two, layout="circo") # plot

#-----#
# HMM Dice Rolling Example
#-----#

suppressMessages(library(depmixS4))

#-----#
# Function to generate the sequence of rolls
#-----#

# Generate the sequence of rolls
rolls <- function(N) {

  dice.col <- num <- NULL

  # Probabilities for each dice
  red.prob <- rep(1/6,6) # 0
```

```

blue.prob <- c(rep(0.08,5),0.6) # 1
dice <- c(1:6)

# select a die at random
dice.col <- sample(c(0,1), size=1, prob=c(0.5,0.5))

# Perform N rolls
for(i in 1:N) {
  if(dice.col[i]==0) {
    num[i] <- sample(dice, size=1, prob=red.prob)
  } else {
    num[i] <- sample(dice, size=1, prob=blue.prob)}

  # switch die unless it comes up 6
  dice.col[i+1] <- ifelse(num[i]!=6, 1-dice.col[i], dice.col[i])
}

dice.col <- ifelse(dice.col==0,"red","blue")
out <- data.frame(Roll=1:N, "Number"=num, "State"=dice.col[1:N])
return(out)
}

set.seed(823)
N <- 50
rolls.out <- rolls(N)

#-----#
# Fit the HMM
#-----#

HMMfit <- function(RollSeq) {
  # Fit the HMM with depmix
  hmm <- depmix(Number ~ 1, data=RollSeq, nstates=2)
  (hmm.fit <- fit(hmm))
  (summary(hmm.fit))

  # The states can be predicted by estimating their posterior
  states <- posterior(hmm.fit)
  return(states)
}

#-----#
# Fit the HMM
#-----#

HMM.out <- HMMfit(RollSeq = rolls.out)

## Initial state probabilities model
## pr1 pr2
## 1 0
##
## Transition matrix
## toS1 toS2
## fromS1 0.667 0.333

```

```
## fromS2 0.409 0.591
##
## Response parameters
## Resp 1 : gaussian
##      Re1.(Intercept) Re1.sd
## St1          6.000  0.000
## St2          3.045  1.522

#-----#
# Plot the results
#-----#

par(mfrow=c(4,1), mar=c(2,4,2,1))

# Observed Rolls
plot(rolls.out$Number, type="o", lwd=2, ylab="")
# Actual states
stateCols <- as.character(rolls.out$State)
cols <- ifelse(stateCols=="red", "plum2", "lightblue1")
barplot(rep(1,N), col=cols, axes=F)
axis(side= 2, at=0.5, line=-1, labels = "TRUE DICE", tick = F, lwd=2, font=2)

#-----#
# Estimated States from HMM
#-----#

barplot(rep(1,N), col=ifelse(HMM.out$state==2,"plum2","lightblue1"), axes = F
)
axis(side= 2, at=0.5, line=-1, labels = "ESTIMATED DICE", tick = F, lwd=2, font=2)

#-----#
# Posterior probabilities
#-----#

plot(HMM.out$S1, type='l', col="lightblue", lwd=2, ylab="")
points(HMM.out$S2, type='l', col="plum", lwd=2)

#-----#
# Table of results
#-----#

table(rolls.out$State, HMM.out$state)

##
##      1  2
## blue 25 11
## red   3 11

#-----#
# Read in and plot the semeion data
#-----#
semeion.data <- read.table("semeion.data", quote = "\"", comment.char = "")

#-----#
```

```

# Put data in format of (1) digit number (2) 256 binary vals
#-----#

nums <- semeion.data[ , 1:256]
digits <- apply(semeion.data[ , 257:266], 1, function(x) which.max(x) - 1) #
change columns to digit number
semeion <- cbind(digits, nums)

#-----#
# Function to plot the digits
#-----#

digitplot <- function(nums) {
  DigitMat <- matrix(as.numeric(nums), nrow=16, ncol = 16, byrow = T)
  image(t(DigitMat[nrow(DigitMat):1,]), col = c("white", "black"), axes=F)
  box("plot")
}

#-----#
# Randomly sample 24 digits to plot
#-----#

set.seed(1)
digitssample <- sample(1:nrow(semeion), size = 24, replace = F)
par(mfrow=c(4,6), pty='s', mar=c(0.5,0.5,0.5,0.5))
for(i in digitssample) {
  digitplot(semeion[i, -1])
}

#-----#
# Fit an HMM to the semeion data
#-----#

suppressMessages(library(depmixS4))

dat <- semeion
datmat <- as.matrix(dat[, -1])
mixmod <- depmix(datmat~1, data=dat, nstates = 10, family=multinomial("identity"))

set.seed(667)
fitmix <- fit(mixmod)

## converged at iteration 51 with logLik: -239114.3

pos <- posterior(fitmix)
states <- pos$state

#-----#
# change the numbering of states to match
#-----#

change <- states
for(i in 0:9) {
  which.num <- as.numeric(names(which.max(table(states[dat$digits==i]))))

```

```

print(which.num)
change[states==which.num] <- i
}

## [1] 8
## [1] 6
## [1] 3
## [1] 1
## [1] 9
## [1] 10
## [1] 5
## [1] 7
## [1] 4
## [1] 2

#-----#
# Training error
#-----#
mean(change!=dat$digits)

## [1] 0.2655367

#-----#
# Kalman Filter
#-----#

suppressMessages(library(dlm))

#-----#
# Generate the random walk model
#-----#

set.seed(143)
z0 <- rnorm(1, 0, 100)
vt <- rnorm(100,0, 0.8)
wt <- rnorm(100,0,0.1)

# compute all z values
z <- z0 + cumsum(wt)
x <- NULL
for(i in 1:100) {
  x[i] <- z[i]+vt[i]
}

#-----#
# plot the random walk
#-----#
par(mfrow=c(2,1), mar=c(2,2,2,2))
plot(x, type='o', pch=16, xlim=c(0,110))
abline(v=100, col="gray", lty=3)

#-----#
# Use a Kalman filter to forecast
#-----#
buildFun <- function(x) {

```

```

    dlmModPoly(order = 1, dV = exp(x[1]), dW = exp(x[2]), m0 = exp(x[3]), C0 =
exp(x[4]))
}

#-----#
# Determine the parameters
#-----#

fit <- dlmMLE(x, parm = c(0,0,0,0), build=buildFun)
fit

## $par
## [1] -0.487450 -6.377063 5.066921 -13.543237
##
## $value
## [1] 27.92525
##
## $counts
## function gradient
##      84      84
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

dlmWalk <- buildFun(fit$par)
V(dlmWalk)

##           [,1]
## [1,] 0.6141906

W(dlmWalk)

##           [,1]
## [1,] 0.00170011

m0(dlmWalk)

## [1] 158.6849

C0(dlmWalk)

##           [,1]
## [1,] 1.312946e-06

#-----#
# Apply the filter
#-----#

walkFilter <- dlmFilter(y=x, mod = dlmWalk)
fitFilter <- walkFilter$m[-1]
lines(fitFilter, type = 'o', pch = 20, col = "maroon")
# obtain the variance matrices from the SVD
v <- unlist(dlmSvd2var(walkFilter$U.C, walkFilter$D.C))

```

```

pl <- fitFilter + qnorm(0.05, sd = sqrt(v[-1]))
pu <- fitFilter + qnorm(0.95, sd = sqrt(v[-1]))
lines(pl, lty = 2, col = "maroon")
lines(pu, lty = 2, col = "maroon")

#-----#
# Forecast the next 10 time steps
#-----#

WalkForecast <- dlmForecast(walkFilter, nAhead = 10)
sqrtR <- sapply(WalkForecast$R, function(x) sqrt(x[1,1]))
pl.f <- WalkForecast$a[,1] + qnorm(0.05, sd = sqrtR)
pu.f <- WalkForecast$a[,1] + qnorm(0.95, sd = sqrtR)

#-----#
# plot the random walk
#-----#

plot(x, type='o', xlim=c(0,110), pch=16)
lines(fitFilter, type = 'o', pch = 20, col = "maroon")
lines(pl, lty = 2, col = "maroon")
lines(pu, lty = 2, col = "maroon")

points(c(101:110), WalkForecast$a, type='o', pch=20, col="slateblue")
points(c(101:110), pl.f, type='l', lty=2, col="lightslateblue")
points(c(101:110), pu.f, type='l', lty=2, col="lightslateblue")
abline(v=100, col="gray", lty=3)

#-----#
# Robot Localization
#-----#
#https://github.com/ioarun/pygame-robotics/blob/master/particle-filter/particle-filter-2.py

set.seed(1212)
par(mfrow=c(2,3), mar=c(4,1,1,1), pty='s')

#-----#
# Move the Robot
#-----#

Step <- function(Point, turn, forward) {

  x <- Point$x
  y <- Point$y

  direction <- turn + rnorm(1, 0, sd=0.1)
  direction <- direction %% (2*pi)

  dist <- forward + rnorm(1, mean=0, sd=0.1)
  x <- x + dist*cos(direction)
  y <- y - dist*sin(direction)

```

```

    return(list(x=x, y=y))
}

#-----#
# Function to sense the surroundings
#-----#

sense <- function(Point, Obstacles) {

  x <- Point$x
  y <- Point$y

  Z <- NULL
  for(i in 1:nrow(Obstacles)) {
    bearing_angle <- atan2(Obstacles[i,1]-y, Obstacles[i,2]-x) + rnorm(1,0,0.
1)
    # Avoid angles greater than 2pi
    bearing_angle <- bearing_angle%%(2*pi)
    Z[i] <- bearing_angle
  }
  return(Z)
}

#-----#
# Measurement function
#-----#

measurent_prob <- function(Point, measurements, Obstacles) {

  # calculate the correct measurement
  predicted_measurements <- sense(Point, Obstacles)

  # compute errors
  error <- 1
  for(i in 1:length(measurements)) {
    error_bearing <- abs(measurements[i]-predicted_measurements[i])
    error_bearing <- (error_bearing+pi)%(2*pi)-pi # truncate

    # update gaussian
    error <- error*dnorm(error_bearing, mean=0, sd=0.1)
  }
  return(error)
}

#-----#
# Draw the particles
#-----#

draw_particles <- function(particles) {
  for(i in 1:nrow(particles)) {
    x <- particles[i,1]
    y <- particles[i,2]
    points(x,y, pch=16, col="lightblue")
  }
}

```



```

    }
  }

#-----#
# Position of the obstacles
#-----#

ObstaclePos <- matrix(sample(x=seq(from=0,to=10, by=0.5), size=20), ncol=2)

#-----#
# create particles
#-----#

particles <- matrix(0, nrow=1000, ncol=2)
for(i in 1:1000) {
  particles[i,1] <- runif(1, 0, 10)
  particles[i,2] <- runif(1, 0, 10)
}

particles <- as.data.frame(particles)
colnames(particles) <- c("x","y")

#-----#
# Draw the region
#-----#

Robot <- data.frame(x=2,y=2)

# Draw the Obstacles and initial Robot position
plot(0, type='n', xlim=c(0, 10), ylim=c(0, 10), xaxt="n", yaxt="n",
      ylab="", xlab=paste0("Time Step: 0"))
draw_particles(particles)
points(ObstaclePos, xlim=c(0, 10), ylim=c(0, 10), lwd=10)
points(Robot$x, Robot$y, pch=8, col="maroon", cex=3, lwd=2)

#-----#
# steps that the robot will take
#-----#

steps <- matrix(c(-1,2,
                  0,2,
                  -1.5,1,
                  -1.5,2,
                  -0.5,2), ncol=2, byrow=T)

#-----#
# Move!
#-----#

ROBOTPos <- as.numeric(Robot)

for(s in 1:nrow(steps)) {
  angle <- steps[s,1]

```

```

step.size <- steps[s,2]

# move the Robot and plot new position
RobotMove <- Step(Robot, angle, step.size)
ROBOTPos <- rbind(ROBOTPos, as.numeric(RobotMove))

p2 <- matrix(0, nrow=nrow(particles), ncol=ncol(particles))
for(p in 1:nrow(particles)) {
  pmove <- Step(particles[p,],angle, step.size)
  p2[p,1] <- pmove$x
  p2[p,2] <- pmove$y
}
particles <- p2
particles <- as.data.frame(particles)
colnames(particles) <- c("x","y")

# Sense the measurements
measurements <- sense(RobotMove, ObstaclePos)

weights <- NULL
for(w in 1:1000) {
  weights[w] <- meausrent_prob(particles[w,], measurements, ObstaclePos)
}

# resampling
particleSample <- sample(1:1000, size=1000, replace=T, prob=weights)
particles <- particles[particleSample,]

# plot the resampled particles
plot(0, type='n', xlim=c(0, 10), ylim=c(0, 10), xaxt="n", yaxt="n",
      ylab="", xlab=paste0("Time Step: ", s))
lines(ROBOTPos, lwd=2, lty=2, col="gray", type='o')
draw_particles(particles)
points(RobotMove$x, RobotMove$y, pch=8, col="maroon", cex=3, lwd=2)
points(ObstaclePos, xlim=c(0, 10), ylim=c(0, 10), lwd=10)

Robot <- RobotMove
}

#-----#
# Markov Random field - Ising Model
#-----#

suppressMessages(library(EBImage))

#-----#
# Plot the original image in black and white
#-----#

image <- readImage("Child.png")
ImageD <- imageData(image)[,,1]

```

```

# change to binary
ImageD <- ifelse(ImageD < 0.9, -1, 1)
dims <- dim(ImageD)

# Plot the original image
x <- 1:dims[1]
y <- dims[2]:1
grid <- expand.grid(x,y)
plot(grid, pch=15, col=ifelse(ImageD==1, "black", "white"), axes=F, asp=1)

#-----#
# Energy function
#-----#

En <- function(x,y,beta=1, eta=1, h=0) {
  # energy for state 1
  x1 <- c(1,x)
  en1 <- exp(-h*sum(x1)-beta*sum(outer(x1,x1,"*"))-eta*sum(x1*y))

  # energy for state -1
  x2 <- c(-1,x)
  en2 <- exp(-h*sum(x2)-beta*sum(outer(x2,x2,"*"))-eta*sum(x2*y))

  return(list(en1,en2))
}

#-----#
# Get the neighbouring nodes
#-----#

neighbours <- function(pos, Mat) {
  # need to obtain the neighbouring nodes, but consider edges and corners
  dims <- dim(Mat)

  # 4 corners
  # top left
  if(all(pos==c(1,1))) {
    x <- c(Mat[1,2], Mat[2,1], Mat[2,2])
    return(x)
  }

  # top right
  if(all(pos==c(1,dims[2]))) {
    x <- c(Mat[1,dims[2]-1], Mat[2,dims[2]-1], Mat[2, dims[2]])
    return(x)
  }

  # bottom left
  if(all(pos==c(dims[1], 1))) {
    x <- c(Mat[dims[1]-1,1], Mat[dims[1]-1,2], Mat[dims[1], 2])
    return(x)
  }

  # bottom right

```

```

if(all(pos==dims)) {
  x <- c(Mat[dims[1], dims[2]-1], Mat[dims[1]-1, dims[2]-1], Mat[dims[1]-1,
dims[2]])
  return(x)
}

# Top row, not a corner
if((pos[1]-1 == 0) & !any(pos[2]==c(1, dims[2])))) {
  x <- c(Mat[pos[1], pos[2]-1], Mat[pos[1]+1, pos[2]-1],
        Mat[pos[1]+1, pos[2]], Mat[pos[1]+1, pos[2]+1],
        Mat[pos[1], pos[2]+1])
  return(x)
}

# Left column, not a corner
if((pos[2]-1 == 0) & !(any(pos[1]==c(1, dims[1])))) {
  x <- c(Mat[pos[1]-1, pos[2]], Mat[pos[1]-1, pos[2]+1],
        Mat[pos[1], pos[2]+1], Mat[pos[1]+1, pos[2]+1],
        Mat[pos[1]+1, pos[2]])
  return(x)
}

# Right column, not a corner
if((pos[2] == dims[2]) & !(any(pos[1]==c(1, dims[1])))) {
  x <- c(Mat[pos[1]-1, pos[2]], Mat[pos[1]-1, pos[2]-1],
        Mat[pos[1], pos[2]-1], Mat[pos[1]+1, pos[2]],
        Mat[pos[1]+1, pos[2]-1])
  return(x)
}

# Bottom row, not a corner
if((pos[1] == dims[1]) & !(any(pos[2]==c(1, dims[2])))) {
  x <- c(Mat[pos[1], pos[2]-1], Mat[pos[1]-1, pos[2]-1],
        Mat[pos[1]-1, pos[2]], Mat[pos[1]-1, pos[2]+1],
        Mat[pos[1], pos[2]+1])
  return(x)
}

# Any other point has all 8 neighbours
x <- c(Mat[pos[1]-1, pos[2]-1], Mat[pos[1]-1, pos[2]],
      Mat[pos[1]-1, pos[2]+1], Mat[pos[1], pos[2]+1],
      Mat[pos[1]+1, pos[2]+1], Mat[pos[1]+1, pos[2]],
      Mat[pos[1]+1, pos[2]-1], Mat[pos[1], pos[2]-1])
return(x)
}

#-----#
# add noise to the image
#-----#

make.noisy <- function(noise=0.1) {
  set.seed(1212)
  NewMat <- ImageD

```

```

for(i in 1:nrow(ImageD)) {
  for(j in 1:ncol(ImageD)) {
    vals <- c(ImageD[i,j], ifelse(ImageD[i,j]==1, -1,1))
    switch <- sample(vals, size=1, prob=c(1-noise,noise))
    NewMat[i,j] <- switch
  }
}

# Plot the new image with noise
plot(grid, pch=15, col=ifelse(NewMat==1, "white","black"), axes=F, asp=1)
return(NewMat)
}

noise.10 <- make.noisy(noise=0.1)
noise.20 <- make.noisy(noise=0.2)

#-----#
# Iterated Conditional Modes algorithm
#-----#

ICM <- function(NoiseMat) {

  # All pixels must be visited at least once
  dims <- dim(NoiseMat)
  Zero <- matrix(0, nrow=dims[1], ncol=dims[2])

  # Initial position is all nodes == observed
  newMat <- NoiseMat

  # Iterate through the pixel matrix
  for(i in 1:dims[1]) {
    for(j in 1:dims[2]) {
      x <- neighbours(pos = c(i,j), newMat)
      y <- c(NoiseMat[i,j], neighbours(pos = c(i,j), Mat=NoiseMat))
      Energy <- En(x,y)
      newMat[i,j] <- ifelse(which.min(Energy)==1, 1, -1) # switch to minimum
energy
    }
  }

  # Plot the denoised image
  plot(grid, pch=15, col=ifelse(newMat==1, "white","black"), axes=F, asp=1)
}

ICM(noise.10)
ICM(noise.20)

#-----#
# Conditional random field
#-----#

suppressMessages(library(crfsuite))

```

```

dat <- read.csv("ner_dataset.csv", header = T)
head(dat)

##      Sentence..      Word POS Tag
## 1 Sentence: 1    Thousands NNS  0
## 2              of      IN   0
## 3      demonstrators NNS   0
## 4              have VBP   0
## 5      marched VBN   0
## 6      through IN    0

NER <- dat
colnames(NER)[1] <- c("Sentence")

#-----#
# total of 479595 sentences
#-----#
N <- nrow(dat)

#-----#
# convert all entries to character strings
#-----#

NER$Sentence <- as.character(NER$Sentence)
NER$Word <- as.character(NER$Word)
NER$POS <- as.character(NER$POS)
NER$Tag <- as.character(NER$Tag)

#-----#
# change the first column to sentence ID label
#-----#

ID <- numeric(N)
j <- 1
ID[1] <- 1
for(i in 2:N) {
  ID[i] <- j
  if(NER$Sentence[i]!="") {
    ID[i] <- j+1
    j <- j+1
  }
}

j # total number of sentences
## [1] 47959

#-----#
# Replace this as the sentence ID
#-----#

NER$Sentence <- ID

#-----#
# view the 8th sentence

```

```

#-----#
(s8 <- NER[NER$Sentence==28,])

##      Sentence      Word POS   Tag
## 638         28        In  IN    0
## 639         28      other  JJ    0
## 640         28  violence  NN    0
## 641         28        ,    ,    0
## 642         28      U.S.  NNP B-gpe
## 643         28 officials  NNS    0
## 644         28      said  VBD    0
## 645         28       one   CD    0
## 646         28 American  JJ B-gpe
## 647         28  soldier  NN    0
## 648         28       was  VBD    0
## 649         28   killed  VBN    0
## 650         28    while  IN    0
## 651         28       on   IN    0
## 652         28   patrol  NN    0
## 653         28       in   IN    0
## 654         28  Baghdad  NNP B-geo
## 655         28   Sunday  NNP B-tim
## 656         28        .    .    0

#-----#
# split into training and test set
#-----#

Ntrain <- round(0.7*N)
train <- NER[1:Ntrain,]
test <- NER[(Ntrain+1):N,]

#-----#
# Fit the CRF
#-----#

crffit <- crf(x = train$Word, y = train$Tag, group = train$Sentence)
pred <- predict(crffit, newdata=test$Word, group = test$Sentence)

#-----#
# Test error
#-----#

mean(pred$label!=test$Tag)

## [1] 0.04970229

```